EXPERT SYSTEM SHELLS IN CHEMISTRY: CHIRULE - A CHIRAL CHROMATOGRAPHIC COLUMN SELECTION SYSTEM USING SIMILARITY SEARCHING AND PERSONAL CONSTRUCT THEORY

by

Scott T. Stauffer

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Chemistry

Approved:

Raymond E. Dessy, Chairman

John G. Mason

Gary L. Long

Mark R. Anderson

Harold M. Bell

September 21, 1993

Blacksburg, Virginia

EXPERT SYSTEM SHELLS IN CHEMISTRY: CHIRULE, A CHIRAL CHROMATOGRAPHIC COLUMN SELECTION SYSTEM USING SIMILARITY SEARCHING AND PERSONAL CONSTRUCT THEORY

by

Scott T. Stauffer

Committee Chairman: Raymond E. Dessy

Analytical Chemistry

(ABSTRACT)

The current process of selecting a column to perform a chiral separation can be characterized as more of an art than a science. This dissertation describes CHIRULE, a "chromatographic assistant" to aid in developing chiral separations.

CHIRULE constructs an n-dimensional information space from a large number of known chiral separations by fragmenting the molecules at their chiral center, producing four molecular fragments. Molecular properties are calculated for each of these fragments. The properties are the axes used to place known separations into the information space.

To suggest a column, the target molecule is added to the information space. Similarity property searching is used to select all known separations similar to the target

molecule. The chemistry-based expert system shell CHESS used to develop CHIRULE also includes features such as functional group recognition and automated knowledge extraction techniques based on Personal Construct Theory.

The results suggest that fragment property values are a route to enhanced understanding and improved selection of chiral separation methods.

ACKNOWLEDGMENTS

There are a number of people that have assisted me in completing this work. They can be broken down into two categories: the first supporting the work itself, and the second supporting me in doing the work. Each area will be covered in turn.

I would like to begin by acknowledging the support of Dr. Keith Casserta and Mr. R. W. Boggs of the Proctor and Gamble Company. Their generous funding of KEY and the early stages of CHESS helped to nurture my initial ideas concerning the project.

Dr. Suzanne Johnson and Mr. John Curruthers of Intel's Components Research invited me for a co-op experience that allowed me to further expand and test many of my ideas concerning computer integration. In addition, they provided funding to continue my work on CHESS.

Intel's Academic Relations deserves my gratitude for the generous equipment grant that provided both a powerful workstation for my research and equipment that enabled our group to set up the network that was later expanded into a departmental network. Susan Langford and Alisa Lane Hampton of Academic Relations provided much help in the process of completing the required documentation.

I owe my thanks to Samuel W. Karickhoff, research chemist at EPA's Athens Environmental Research Laboratory and Lionel Carriera, Professor of Chemistry at the University of Georgia. They graciously consented to provide me with a beta copy of SPARC, the molecular property estimator utilized in this work. This key element was vital to the success of CHESS.

Dr. William E. Weiser, a development scientist at the Burroughs Wellcome Company, first introduced me to chiral chromatography during his lecture for Industrial Instrumental Analysis. I am thankful that our paths crossed long enough for that seed to be sown. The success of CHIRULE indicates that chiral chromatography was in fact a fertile area for expert systems technology to take root and grow.

Dr. John W. Roach, Associate Professor of Computer Science at Virginia Tech, deserves considerable credit for the patient tutoring he provided in the areas of artificial intelligence and expert systems. His guidance during the formative stages of the research were invaluable.

I am indebted to the all of the members, both past and present, of the Laboratory Automation and Instrument Design (LAID) group. The lessons learned, the character building experiences endured, and especially the camaraderie will be sorely missed.

A special thanks goes to Dr. Ray and Lee Dessy for their guidance during my maturation process. I cannot think of a better environment in which to have obtained my degree. Going against the cookie-cutter approach now so prevalent to generating supposed professionals, he has provided his research group with a fertile environment in which to envision and develop their own ideas. I can only hope to yet again prove his methods valid by following in the footsteps of so many of his former students.

Many thanks to my parents, John & Evelyn Stauffer, for their continuous love and support. To Mom for constantly reminding me that I CAN do it. And to Dad for his at times annoying saying that I now think I understand: "The man that know's how can always find a job working for the man that knows why..."

And lastly, but certainly not the least, to my best friend, loving mother of our child Sarah Marie, and beautiful wife. Without her understanding, love, and unyielding support I could have never completed this work. I am indebted to her forever, and look forward to a lifetime of attempting to repay her kindness.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGMENTS iv

TABLE OF CONTENTS vii

LIST OF FIGURES x

LIST OF TABLES xii

LIST OF TEXT FRAMES xiii

I. Introduction 1

II. Historical 12
  II.A. Chromatography 12
    A.1. Chiral Chromatography 12
    A.2. Three-Point Rule 16
    A.3. Types of CSPs 18
    A.4. Chromatographic Expert Systems 24
    A.5. Predicting Chiral Separations 32
  II.B. Chemistry-Based Expert System Shell 37
    B.1. Molecular Representation 38
    B.2. Functional Group Recognition 41
    B.3. Property Estimation 45
    B.4. Similarity Searching 48
    B.5. Automated Knowledge Acquisition 51
    B.6. Expert Systems 62
    B.7. Expert System Languages & Shells 71

III. CHESS's General Chemical Knowledge 79
  III.A. Molecular Representation 82
  III.B. Functional Group Recognition 86
  III.C. SPARC and Property Estimation 95

IV. Automated Knowledge Acquisition - KEY 99
  IV.A. Choosing an Appropriate Topic 101
  IV.B. Example - OPTIMER 102
  IV.C. Grid Acquisition 116
  IV.D. Grid Analysis 121
  IV.E. Knowledge Base Generation 134
  IV.F. Data Structures and Files 141

V. Chiral Chromatography and CHIRULE 150
  V.A. EXAMPLE - CHIRULE 152
  V.B. Molecular Properties 154

B.1. Polarizability Volume 156
B.2. Hydrogen Bonding 158
B.3. Molecular Volume 161
V.C. Similarity Searching 164
C.1. Hashing Data Sets 167
C.2. CHIRULE's Hashing Function 170
C.3. CHIRULE's Similarity Search 174
3.a. Overview 175
3.b. Implementation 176
C.4. Generating Column Suggestions 180
4.a. Identical Fragments 181
4.b. Property Normalization 182
4.c. Permutations 184
V.D. Extending CHIRULE 188
D.1. Adding Properties 188
1.a. Changes to CHIRULE 189
1.b. Interactions With SPARC 191
1.c. Knowledge Bases 192
D.2. Chiral Separation Databases 193
D.3. Property Normalization Factors 194

VI. Prolog 195
VI.A. Programming in Prolog 196
VI.B. Automatic Backtracking 205
VI.C. Relentless Searching 206
VI.D. Strategy for Loading Prolog Rulebases . . . 208
VI.E. Ring Determination Within Molecules 210

VII. System Overview 213
VII.A. System Technical Notes 214
A.1. Computer Workstation 214
A.2. SPARC 215
A.3. MACCS and ChemTalk 218
A.4. KEY 221
A.5. Turbo-Prolog and KEY 223
A.6. VPI Prolog and CHESS 225
VII.B. Knowledge Bases 226
VII.C. Network Integration and Information Flow . 234
C.1. KEY Development 235
C.2. Prolog Code Development 237
C.3. CHIRULE Development 239
VII.D. Loading Rules and Running Systems 246
D.1. CHESS 247
D.2. OPTIMER and KEY 248
D.3. CHIRULE 252

VIII. Results and Discussion 261
VIII.A. CHIRULE's Results 263
A.1. Discussion of CHIRULE's Results 263

A.2. CHIRULE's Column Suggestion Methodology 285
A.3. Property Adjustment and Delta Values . . 287
A.4. CHIRULE's Total Difference Measure . . . 288
A.5. CHIRULE's Testing Procedures 289
5.a. Testing by Molecule 291
5.b. Testing by Column 292
5.c. Other Testing Procedures 293
A.6. Future Directions 294
VIII.B. OPTIMER Results 295
B.1. Discussion of Results 296
B.2. Testing Procedures 305
B.3. Future Directions 305
VIII.C. System Integration Issues 310
VIII.D. A Concluding Analysis 311

IX. Conclusion 313

X. Addendum 317

REFERENCES 327

VITA 339

VOLUME II - APPENDICES 340
APPENDIX A: Dialog of a Session Using KEY 341
APPENDIX B: OPTIMER Rulebase Generated by KEY . . 352
APPENDIX C: KEY Source Code 354
APPENDIX D: KEY Output File Formats 386
APPENDIX E: KEY Inference Engine Code 388
APPENDIX F: CHESS Source Code 395
APPENDIX G: Knowledge Bases Used by CHESS 415
APPENDIX H: OPTIMER Testing Results 425
APPENDIX I: CHIRULE Testing Results 442
APPENDIX J: Daicel Chiral Column Selection Guide . 493

LIST OF FIGURES

Figure 1: CHESS Shell Overview 4

Figure 2: Network Integration 8

Figure 3: Dalgliesh Three Point Interaction 17

Figure 4: ESCA Project Domain 27

Figure 5: Similarity Searching Targets 42

Figure 6: Differences Between CHIRULE and OPTIMER . . . 81

Figure 7: Daicel Chiral Column Selection Guide 103

Figure 8: Grid Data Structure 142

Figure 9: Match List Data Structure 143

Figure 10: Similarity Searching Procedure 165

Figure 11: Hashing Functions 168

Figure 12: Structures for Molecular Formula ($C_8H_9NO_2$) . 199

Figure 13: CHESS and CHIRULE Knowledge Bases 227

Figure 14: KEY Development Strategy 236

Figure 15: Prolog Development Environment 238

Figure 16: CHIRULE Development Environment 240

Figure 17: Using CHIRULE's Development Environment . . 253

Figure 18: CHIRULE Results for CrownPak CR, Group 1 . . 265

Figure 19: CHIRULE Results for CrownPak CR, Group 2 . . 266

Figure 20: CHIRULE Results for CrownPak CR, Group 3 . . 267

Figure 21: CHIRULE Results for CrownPak CR, Group 4 . . 268

Figure 22: CHIRULE Results for ChiralCel OB, Group 1 . 269

Figure 23: CHIRULE Results for ChiralCel OB, Group 2 . 270

Figure 24: CHIRULE Results for ChiralCel OB, Group 3 . 271
Figure 25: CHIRULE Results for ChiralCel OD, Group 1 . 272
Figure 26: CHIRULE Results for ChiralCel OD, Group 2 . 273
Figure 27: CHIRULE Results for ChiralCel OD, Group 3 . 274
Figure 28: CHIRULE Results for ChiralPak WH, Group 1 . 275
Figure 29: CHIRULE Results for ChiralPak WH, Group 2 . 276
Figure 30: CHIRULE Results for ChiralPak WH, Group 3 . 277
Figure 31: Investigating CHIRULE Test Data 282
Figure 32: OPTIMER Results, Group 1 297
Figure 33: OPTIMER Results, Group 2 298
Figure 34: OPTIMER Results, Group 3 299
Figure 35: OPTIMER Results, Group 4 300
Figure 36: OPTIMER Results, Group 5 301
Figure 37: OPTIMER Results, Group 6 302
Figure 38: OPTIMER Results, Group 7 303
Figure 39: OPTIMER Results, Group 8 304
Figure 40: Testing Scheme for CHIRULE 322

LIST OF TABLES

Table I: Acronyms Utilized in Describing CHESS 10

Table II: Descriptions of CSP Types 19

Table III: CHESS Functional Group "Hook" Queries . . . 87

Table IV: Functional Group Central Atom* Definitions . 89

Table V: Functional Group Set II Definitions 90

Table VI: Atom and Bond Determination Predicates . . . 94

Table VII: Comparison of SPARC Polarizability Values . 159

Table VIII: Comparison of SPARC Molecular Volume Values 162

Table IX: Similarity Searching Definitions 166

Table X: Steps for Adding Properties to CHIRULE 190

Table XI: CHESS and CHIRULE Knowledge Bases 228

Table XII: CHESS Program Modules 249

Table XIII: CHIRULE Testing Results 264

Table XIV: CHIRULE Single Property Results 283

Table XV: CHIRULE Testing Predicates 290

Table XVI: CHIRULE/Chirbase Results 319

Table XVII: Molecules used in Chirbase Experiment . . . 321

LIST OF TEXT FRAMES

Text Frame 1: Guide to Readers 11

Text Frame 2: MOL File of Phenylglycine 83

Text Frame 3: Sample SMILES Strings 84

Text Frame 4: SPARC MOL File Format 84

Text Frame 5: Hydrogen Deficient MOL File Input for CONVB 85

Text Frame 6: Using Functional Group Predicates 91

Text Frame 7: Predicate for Determining Carbonyls . . . 92

Text Frame 8: SPARC Batch Mode Input File 96

Text Frame 9: SPARC Batch Mode Output File 96

Text Frame 10: SPARC Dribble File 98

Text Frame 11: Spreadsheet Format of Grid 105

Text Frame 12: Element Entry Menu 105

Text Frame 13: Entering OPTIMER Elements 106

Text Frame 14: OPTIMER Element List 106

Text Frame 15: Entering Traits Using Triads of Elements 107

Text Frame 16: OPTIMER Trait List 108

Text Frame 17: Entering Grid Values 109

Text Frame 18: Optimer Grid 109

Text Frame 19: Element Cluster Analysis 110

Text Frame 20: Trait Cluster Analysis 111

Text Frame 21: Assigning Trait Weights 112

Text Frame 22: OPTIMER Grid With Trait Weights 112
Text Frame 23: Generating HC Prolog Rulebase 113
Text Frame 24: OPTIMER Rulebase Format 114
Text Frame 25: Choosing a Molecule for OPTIMER 114
Text Frame 26: KEY Using "Hooks" Built into CHESS . . . 115
Text Frame 27: KEY Querying User for Trait Value . . . 115
Text Frame 28: OPTIMER Results 116
Text Frame 29: KEY Screen Size Selection 117
Text Frame 30: Getting a Saved Element List 118
Text Frame 31: Saving KEY Element Lists 118
Text Frame 32: Adding to the Trait List 119
Text Frame 33: Grid Analysis Menu 121
Text Frame 34: Element Analysis Menu 122
Text Frame 35: Element Correlation Analysis 123
Text Frame 36: Trait Analysis Menu 125
Text Frame 37: Trait Correlation Analysis 126
Text Frame 38: Grid Element Edit Menu 127
Text Frame 39: Adding Elements to the Grid 128
Text Frame 40: Changing a Grid Element 129
Text Frame 41: Changed OPTIMER Grid 129
Text Frame 42: Element Cluster Analysis Showing
Inconsistency 130
Text Frame 43: Automated Analysis Uncovering Match . . 131
Text Frame 44: Similarity Resolution Menu 131
Text Frame 45: Verifying an Element 132

Text Frame 46: Combining Two Elements 132
Text Frame 47: Rule Generation Menu 135
Text Frame 48: Running a KEY Rulebase 138
Text Frame 49: OPTIMER Trait Data 140
Text Frame 50: Grid Data Structure 144
Text Frame 51: Match List Data Structure 147
Text Frame 52: KEY Element File 147
Text Frame 53: KEY Trait and Grid Files 148
Text Frame 54: Property Adjustment Values 172
Text Frame 55: Property Normalization Values 183
Text Frame 56: Index of Hydrogen Deficiency Examples . 196
Text Frame 57: Prolog Facts 197
Text Frame 58: Prolog Predicates 198
Text Frame 59: Prolog Queries 200
Text Frame 60: Predicate "count_num" 203
Text Frame 61: Relentless Searching 206
Text Frame 62: Calculating the Number of Rings Within a Molecule 211
Text Frame 63: Logging onto VTCC1 using the IBM Rolm System 219
Text Frame 64: MACCS Logon Procedure 220
Text Frame 65: Molecule Knowledge Base 229
Text Frame 66: Molecular Fragment Knowledge Base . . . 230
Text Frame 67: Fragment Value Knowledge Base 231
Text Frame 68: Chiral Column Knowledge Base 231

Text Frame 69: DO-CONVB.BAT 242
Text Frame 70: CONVBALL.BAT 243
Text Frame 71: TO-SPARC.BAT 244
Text Frame 72: SPARCALL.WPM 245
Text Frame 73: MOL File Name for CHIRULE 255
Text Frame 74: Generating SMILES String for CHIRULE . . 256
Text Frame 75: Generating Fragment SMILES Strings for CHIRULE 257
Text Frame 76: Generating SPARC Fragment Values for CHIRULE 258
Text Frame 77: CHIRULE Column Suggestions 259
Text Frame 78: CHIRULE Adding Molecule to Knowledge Base 260
Text Frame 79: Unexpected OPTIMER Result 296
Text Frame 80: Prolog Predicate for Determining Amides 306
Text Frame 81: Color Rulebase 307
Text Frame 82: Automobile Rulebase 308

I. Introduction

The separation of biologically active compounds has become a very prominent field of research in the past several years. This is of particular importance in light of the recent FDA policy clarification on chiral drugs[1]. The capability to resolve enantiomers in pharmaceutical, pesticides, and other biological compounds has provided many challenges to chromatographers. The work described herein is an effort to provide scientists with a "chromatographic assistant" to aid in developing chiral separations. This computer based assistant, called CHIRULE, utilizes expert systems technology to help the chromatographer make an initial choice of a chiral column to use in developing a separation method. Both CHIRULE and CHESS, the tool set used to build CHIRULE, were developed by the author.

There have been a number of computer systems developed to suggest columns and conditions for chromatographic separations. Most of them rely heavily on chromatographic theory, and work well when dealing with even marginally dissimilar compounds. However, when it comes to resolving racemic mixtures, the theory governing these separations has not yet been fully developed, and earlier systems do not provide answers. Developing such a system for chiral separations requires that a heavy dependence be placed on

currently existing data, and that similarities between the target molecule and the existing knowledge base be found.

The mechanism used by CHIRULE to suggest a column for the separation of enantiomers is based on the chemical environment around the chiral center. Chiral centers consisting of four substituents, as opposed to an axis of chirality, are addressed. The molecule is split into four molecular fragments by putting a hydrogen atom in place of the bond to the chiral carbon atom. Calculations on each of the four fragments are then performed for several properties, such as molecular polarizabilities, molecular volumes, and hydrogen bonding. Each of the properties address a different aspect of the chemical environment experienced by the chiral carbon.

Known separation methods are used to create an knowledge base of known chiral column separations. This knowledge base is an n-dimensional information space, where each dimension represents one of the properties. Each point in the information space represents a chiral molecule with a known separation method. When an unknown or target molecule is posed to the system, its properties are calculated and it is placed within the information space. Molecules that fall within a certain distance to the target molecule are considered to be chemically similar since their chiral carbons experience similar chemical environments. A chiral

column is then suggested for the target molecule based on chemically similar molecules found within the chiral knowledge base.

CHIRULE was developed using CHESS, a CHemistry-based Expert System Shell, also described herein (see Figure 1). The goal of CHESS is to introduce expert systems into the field of chemistry by enabling scientists to develop their own expert system applications. The more important capabilities provided to the user include an automated knowledge extraction utility, a molecular property evaluator, functional group recognition, and a chemical structure manipulator.

There are two different perspectives to expert systems: that of the developer and that of the application user. CHESS is a tool used to develop expert system applications. It is a collection of programs utilizing various languages that allow an expert to condense knowledge into rules and organize those rules into a knowledge-based system. CHIRULE is an application that was produced by a scientist, who has a working knowledge of expert systems, using the expert system shell CHESS. An application user would run CHIRULE to suggest what chiral column to use to separate a racemic mixture.

Traditionally, a scientist that was interested in developing an expert system application would have to

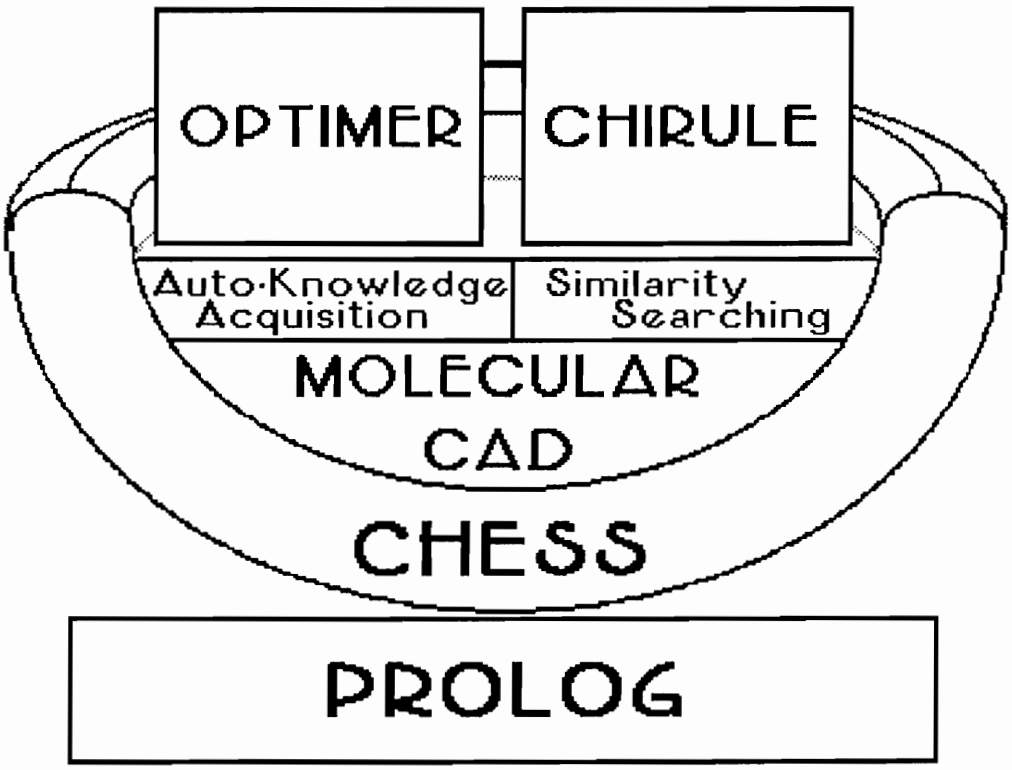


Figure 1: CHESS Shell Overview

consult with a knowledge engineer (a computer scientist), who would actually do the coding of the information. There would be many barriers to doing this since each spoke their own language, each of which was un-intelligible to the other. Such efforts often resulted in systems which did not live up to either's expectations, and sometimes even ended in outright failure. The goal is to introduce CHESS to enable application users, specifically scientists, to become expert system developers themselves. By providing a tool that scientists can experiment with by themselves, they will be able to construct their own expert systems. In doing this, it is hoped that they can become somewhat familiar with the technology and visualize additional areas where it can be applied. Then, if an initial application shows promise and a fully developed system is warranted, some of the language barriers between the scientist and knowledge engineer will have already been bridged, greatly increasing the chances of success.

A major part of CHESS is KEY, an automated Knowledge Extraction utilitY. With KEY, the scientist can interact with a computer to develop an expert system application without having to become a computer scientist or requiring the help of a knowledge engineer. Once an initial system is input into KEY, the knowledge base is generated and then checked for ambiguous and incomplete information.

Corrections can be made either automatically or manually. OPTIMER is another application constructed using CHESS and KEY (see Figure 1). It serves as an example of how a scientist, not familiar with expert systems techniques or programming, can develop an application that serves as a replacement for a classical printed application guide for chiral column selection. OPTIMER assists in selecting a Chiral HPLC column for Optical Isomer Resolution.

In order for CHESS to recognize molecules, a graphical molecule drawing package is needed whose data is then used as input. MACCS and ISIS are molecular drawing programs from Molecular Design Limited (MDL) that run on DEC VAX and other hardware platforms. It has the capability not only to interactively draw molecules on the screen, but also to create databases of molecules and perform searches on them. Molecules are saved to MOL files, which can then be read into the CHESS system. The MOL file format is now available publicly, and is used by many vendors.

In building an expert system, it may be necessary to base a decision on a particular property of a compound. CHESS adds the capability of property estimation by utilizing SPARC, a program developed by the University of Georgia for the EPA. SPARC is an acronym for SPARC Performs Automated Reasoning in Chemistry. SPARC calculates over a dozen properties based on molecular structure. CHESS

outputs the appropriate information, and then reads in the results generated by the SPARC system.

Another important capability of CHESS is its ability to reason on chemical structures. Once the MOL file for a molecule is read into the system, it has the capability not only to determine if particular atoms or bonds are present, but also to base decisions on the location of functional groups within a molecule.

All of the modules that have been included in CHESS have been chosen because of their capabilities without regard to the computer platform on which they run (see Figure 2). Each module runs on a different computer system, with a single DOS based workstation providing a common access to each of them. This seamless integration is provided by way of a university wide computer network.

In developing an application, a scientist would begin building the application using KEY. Through KEY, the scientist would have access to all the "hooks" built into CHESS. The "hooks", or capabilities of CHESS, include molecular structure manipulation, functional group recognition, and property estimation. After completing several levels of revisions to refine the knowledge base, experimentation with the newly developed expert system could proceed. Once the process of refinement and testing is

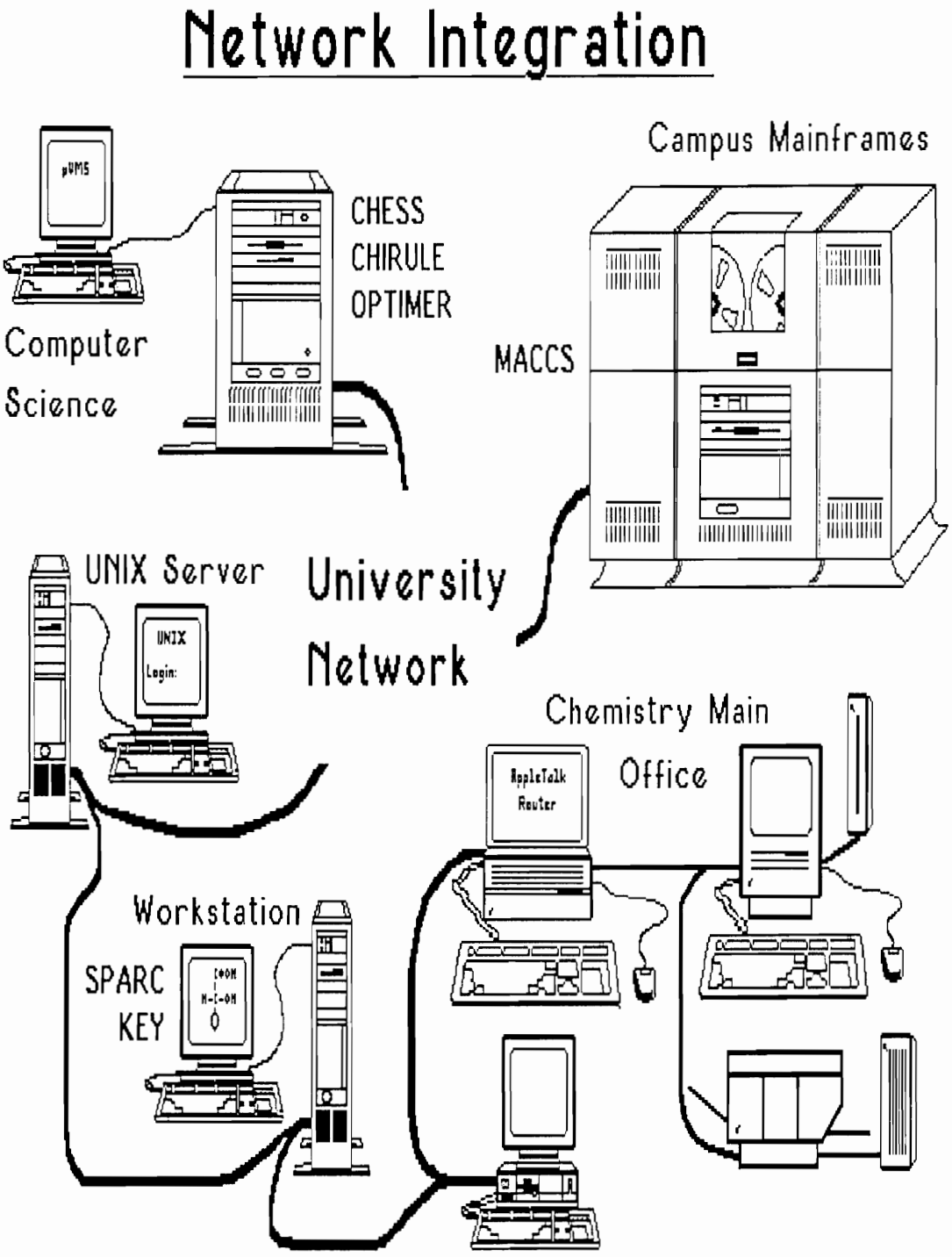


Figure 2: Network Integration

completed, the system would be ready for use, solving real problems for scientists. Once the scientist was convinced of the possible application of expert systems to their work, the initial system could be expanded into a larger system using the capabilities of CHESS and Prolog, the language in which CHESS is written, to their fullest extent.

Through the use of CHESS and all of the capabilities available within it, scientists will be able to not only begin to understand expert systems technology, but also see how expert systems are applicable to their own individual areas of science. This will provide a new tool, previously un-available to scientists, that they can use in their own areas of research. Both CHIRULE and OPTIMER are chemical examples of CHESS's utility in constructing expert systems.

There are a number of different modules developed and described herein, most of which have already been introduced above. Because each module is distinct, and understanding the differences between them is imperative to perceiving their contributions, each module is explained in Table I. It is suggested that the reader refer back to this table often while perusing this work.

Readers of this dissertation are likely to come from diverse backgrounds in the areas of chemistry and computer science. Text Frame 1, in conjunction with the Table of Contents, will prove invaluable in guiding readers to the

Table I: Acronyms utilized in describing CHESS

CHIRULE - An expert system application to assist chemists in suggesting a chiral column to begin a method development to separate enantiomers.

CHESS - (CHemistry-based Expert System Shell) A collection of tools used to develop chemistry related expert systems.

KEY - (Knowledge Extraction utilitY) An automated knowledge extraction utility that automatically develops expert systems using directed interaction with an expert.

PCT - (Personal Construct Theory) The theory upon which KEY is developed.

OPTIMER - An example of an expert system constructed by a chemist using KEY which codifies the decision tree knowledge found in most commercial chiral column selection guides.

MACCS, ISIS - Software packages developed by Molecular Design Limited (MDL) used to generate MOL files that enabled input of molecules into CHESS.

MOL file - A textual representation of a molecule utilizing connectivity tables.

SPARC - (SPARC Performs Automated Reasoning in Chemistry) A program that calculates various molecular properties based only on molecular structure.

User - A Scientist that wants to use expert systems.

Developer - A Scientist that uses CHESS or KEY to develop their own expert system.

Target Molecule - The molecule for which a Scientist needs to develop a chromatographic method.

Sim-List Molecule - A molecule chosen from the Molecule KB that the similarity search uses to compare to the target molecule.

KB - An abbreviation for Knowledge Base (see Chapter VII.B).

particular sections of interest to them. It is suggested that the reader begin investigating Chapter II, where each section is intended to give the reader a general introduction to one of the fundamental topic areas. From there, the reader is directed to the individual section in later chapters that discusses a particular topic in depth. All readers are directed specifically to sections IV.B and V.A where OPTIMER and CHIRULE are described. These are the two example systems that were constructed using CHESS, the chemistry based expert system shell. This tool is pivotal. There are a number of sections that contain references to material in other chapters. These references are present to (1) assist the reader in completing an understanding of the topic covered in the section being studied, and (2) show how that topic fits into the overall work.

Topic of Interest	Chapter
Chiral Chromatography	II.A, V.A, VIII.A
Similarity Searching	II.B.2, V.C
Automated Knowledge Acquisition	II.B.5, IV, VIII.B
System Integration	VII.C, VII.D, VIII.C
Molecular CADD	II.B.1-3, III
CHESS	I, III, VII.D.1
OPTIMER	I, IV.A, VII.D.2, VIII.B
CHIRULE	I, V, VII.D.3, VIII.A

Text Frame 1: Guide to Readers

II. Historical

The two major areas addressed in this work are chiral chromatography and expert systems. Reviewing the backgrounds of each of these areas is imperative to understand how they can be successfully combined into a single working system.

II.A. Chromatography

Chromatography has proven to be fertile ground to a number of expert system applications. The tools used to construct such systems must understand some basic chemistry principles in order to be successful. Delving into the background of chiral chromatography seems a reasonable place to begin our journey.

II.A.1. Chiral Chromatography

The capability to successfully resolve enantiomers has generated an enormous amount of interest in the past few years, especially from pharmaceutical companies. Enantiomers, or two molecules that are nonsuperimposable mirror images of each other, are also known as chiral (from the Greek word, cheiro, meaning hand [as in a pair of hands]). Chiral chromatography is under investigation for use to remove the unwanted enantiomer from the drugs that

are racemic mixtures (approximately 25% of widely prescribed drugs[2]). In some cases, the unwanted enantiomer is simply an inactive component, and its removal will increase the efficacy of the drug. Conversely, enantiomers can have vastly different therapeutic effects. Dextromethorphan is an over-the-counter antitussive, whereas its stereoisomer levomethorphan is a controlled narcotic[3]. There have been cases where the unwanted enantiomer has proven to be dangerous. Thalidomide (racemic *n*-phthalylglutamic acid imide) was a sedative prescribed in the early 1960's, whose *S*-enantiomer was found to cause serious fetal malformations[4].

While these are extreme examples, further investigation points to the extreme amount of interest recently shown towards chiral chromatography: 12 of the 20 most prescribed drugs in the USA and 114 of the top 200 possess at least one chiral center[5]. Consulting the U.S. Pharmacopeial Dictionary of Drug Names would show that almost half of the 2050 drugs contained therein contain at least one chiral center. In addition, the racemic or diastereomeric forms have been used in 400 of them[6]. This data is further evidence showing that interest in resolving enantiomeric forms of compounds will continue to increase over the coming years. A look back at a number of prominent works in this area is in order.

Pasteur, describing his 1848 work with sodium ammonium tartrate, demonstrated the principles of optical activity in separating the enantiomorphs[7]. The racemic mixture was easily separated by hand, as the enantiomers each had a distinct crystalline shape at temperatures below 27°. Although recognizing the two optical isomers and that optical activity required the molecular dissymmetry resulting from their non-superimposability, he never determined what caused the molecular dissymmetry[8]. Further explanations into the story that lead to Pasteur's discovery have been done elsewhere[9].

A further step in explanation of chirality occurred in 1874 when the concept of an "asymmetric carbon atom" was proposed independently by Van't Hoff[10] and Le Bel[11]. Because of Le Bel's square pyramid geometry of the chiral center, Van't Hoff and his tetrahedral organization generally received greater credit for the discovery.

The first to correctly attribute a separation to an enantioselective process was the English scientist Dalgliesh working with cellulose molecules that were optically active[12]. Dalgliesh is often given credit for stating, although primitively, the structural requirements needed in a column packing material for recognizing the asymmetric center of a molecule. This is referred to the three point

rule for describing the enantioselective interaction between the chiral molecule and the chiral stationary phase.

The late 1970's saw a heightened interest in stereochemistry in many scientific disciplines as capabilities in the areas of regiospecific reactions and enantiomeric control were realized. The introduction of a commercial Chiral Stationary Phase column, or CSP, in 1981 by Pirkle[13] was the first of what is now a flood of columns available from various manufacturers. However, now that there are a large number of CSPs (over 1,000), the new challenge is knowing what column is the correct one for the current application.

In an effort to keep track of all the new CSPs that have been developed and the chiral molecules they have separated, researchers in Europe have developed a database that contains all of the reported separations found in the literature since 1977. Roussel and Piras have developed ChirBase[14], a graphic database for the enantiomeric resolution of racemic mixtures by chiral liquid chromatography, using ChemBase, a chemical database package from MDL[15]. It currently contains over 20,000 entries and a license costs over a dollar per entry.

The capabilities of ChirBase include sub-structure querying, and retrieving and sorting all fields contained in the database. For each entry, the following information is

provided: Molecular structure, complete reference, commercial and chemical names, elution order, capacity factors for both, separation factor, resolution, experimental conditions, CSP and supplier, and general comments about the separation. The capability also exists for creating a separate database containing data on personal separations that have been achieved.

II.A.2. Three-Point Rule

As stated previously, Dalgliesh is usually given credit for primitively stating the three point rule for describing the enantioselective interaction between the chiral molecule and the chiral stationary phase (see Figure 3). Pirkle[16] has restated this classical model as follows:

> Chiral recognition requires a minimum of three simultaneous interactions between the CSP and at least one of the enantiomers, with at least one of these interactions being stereochemically dependant."

It is important to note that the word interactions is used, indicating that either attractive or repulsive forces can effect the separation of enantiomers. Pirkle characterized these interactions as perturbations that occur as two molecules orbitals begin to appreciably overlap. Hydrogen bonding, dipole stacking, and steric interactions are cited

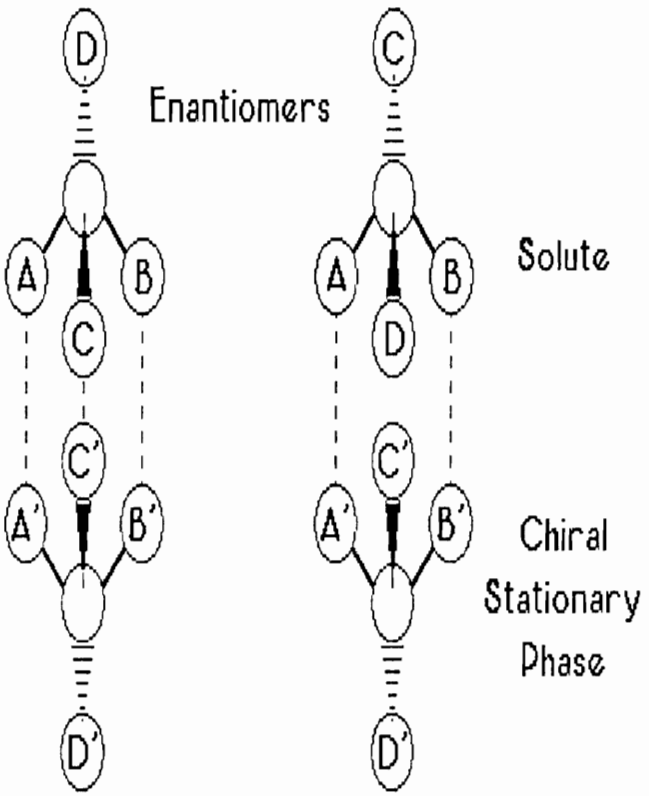


Figure 3: Dalgliesh Three Point Interaction

as specific perturbations resulting from a molecule's electronic orbital energies.

It is by no means certain that this is the correct method of describing the interaction between the CSP and the enantiomers. Other studies have suggested that small through-space field effects provide a more accurate view of the interaction (see II A.5 Predicting Chiral Separations below).

II.A.3. Types of CSPs

In an effort to better understand CSP's, they are frequently divided into categories based roughly upon their separation mechanisms. However, there seems only to be rough agreement upon what those categories should be. Wainer[17,18], Pirkle[16], Armstrong[19,20], Däppen[21], and most recently Taylor and Maher[2] all describe the categories similarly, yet give them different titles. Especially confusing is the use of the terminology "Type 1 - 6" by Taylor and Maher that differs from the "Type I - V" first proposed by Wainer. The classification scheme used herein is based on a description of the interaction, and is roughly based on Wainer's scheme. Table II compares the different methods referenced above to this scheme. For a more complete description of each of the categories, see Taylor and Maher[2].

Table II: Descriptions of CSP Types

Wainer	Pirkle, et.al.	Armstrong,et.al.	Taylor & Maher
:Type I- attractive, H-bonding, π-π, and dipole stacking interactions (i.e. on surface)	:Donor-Acceptor CSP's - H-bonding, π donor-acceptor, dipole stacking	- π Complex, Hydrogen bonding	- Type 3 - Asymmetric Strand packings
:Type II- attractive interactions, but some inclusion (in cavities or ravines)	:Chiral polymers - natural chiral adsorbents such as cellulose or sephadex	:Chiral polymers - either natural (cellulose or polysaccharides) or synthetic (poly-methacrylate)	:Type 5 - high mass Polymer, either natural (cellulose) or synthetic (poly-acrylates, poly-acrylamides)
:Type III-chiral cavities and inclusion complexes	:Host-guest complexation - crown ethers, chiral cavity, cyclodextrines	:Inclusion complexation - cyclodextrins, crown ethers	:Type 4 - Chiral cavity, low molecular mass CSPs, based on inclusion (crown ethers and cyclodextrins)
:Type IV- diastereomeric metal complexes (ligand exchange)	:Ligand exchange chromatography	:Ligand exchange	:Type 1 - Ligand exchange, diastereomeric metal complexes
:Type V- Protein CSP, based on hydrophobic and polar interactions	:Protein-derived CSPs	:Protein bonded phases	:Type 6 - Silica-bonded protein CSPs
:not described	:not described	:Charge transfer	:Type 2 - Charge transfer

Attractive interactions such as hydrogen bonding, π-π interactions, and dipole stacking are the major contributors that result in the separation of isomers on this first type of chiral column. These CSPs are also known as π complex, donor-acceptor, brush type, asymmetric strand, or even Pirkle type columns. Some researchers have sub-divided this category further, denoting π acceptor, π donor, urea π donor, and urea type columns. They often consist of chiral moieties connected to the silica network via a hydrocarbon chain, and arranged in much the same way as the bristles on a brush. Hexane and isopropanol are typical mobile phases used for these normal phase separations. Additives such as acetonitrile or ethanol can be used to increase efficiency and reduce retention times.

A second CSP type is also based on attractive interactions, but where inclusion type complexes play a more important role. It makes use of hydrogen bonding, dipole stacking, and π-π interactions between the solute and the CSP's chiral discriminators located within the cavities of the polymer matrix, rather than on the surface of the silica as described above. This mixture of the first and third CSP types (inclusion) is usually referred to as chiral polymers or helical polymers, and is further divided by some researchers into natural and synthetic polymers. Cellulose, with its helical asymmetric strands, is the most common

natural polymer used as a CSP for this type, while the synthetic polymers based upon polyacrylamides and polyacrylates are also widely used. Alcohol modified hexane or other non-polar mobile phases are most often used. Derivitization may be required to attain the required interactive groups.

Inclusion of the solute within a chiral cavity of the CSP describes the next mechanism. This type is described as chiral cavity packings, cyclodextrin phases, and even inclusion or host-guest complexations. The CSPs are usually made up of either crown ethers or cyclodextrins, and are sometimes even divided into two different types. Solutes usually contain an aromatic moiety near or at the chiral center, and those that most completely fill the chiral cavity are retained longest. Although usually based solely on the hydrophobic interaction of a group that fits within a cavity, the separation can also be effected by the hydrogen bonding to a chiral exterior. Poor band shapes can be a result of the kinetics reflecting the relatively slow process of inclusion, which requires that the solutes must first fit into the cavity before the three point interaction can take place. Both polar and non-polar mobile phases are used.

Ligand exchange chromatography (LEC) involves the reversible formation of a complex between a metal ion and

the chiral complexing agents. A diastereomeric metal complex is formed when the unfilled inner shell d-orbitals of the metal ion accept electrons donated by suitable solute ligands (such as amino or carboxylate groups). Factors such as pH, temperature, and ionic strength of the mobile phase affect the efficiency and selectivity of the separation. It is usually the method of choice for free and underivatized amino acids and amino acid derivatives because of its use of aqueous mobile phases and the lack of a separate derivatization step. However, difficulties can arise in preparative separations in part because of the need to remove trace metal ions. Long retention times and poor band shapes due to unfavorable desorption kinetics are often found with LEC.

The fifth CSP can be classified as an affinity between the protein stationary phase and the solute. This type is usually referred to as protein CSPs, silica-bounded proteins, and sometimes affinity phases. Although the mechanism is not well understood, hydrophobic, polar, and even steric interactions are thought to be important. These CSPs are most attractive for pharmaceutical applications, where protein chiral selectors are well suited for biologically active analytes. CSPs derived from bovine serum albumin (BSA) and human α_1-acid glycoprotein are most

common. Aqueous mobile phases containing phosphate buffers are typical.

The final CSP type is sometimes not mentioned because of its rather limited utility. It is based on immobilizing chiral charge transfer molecules on silica gel. Helicenes have been the only reported use for these CSP's.

The recent explosion of commercial CSPs has made the task of choosing a CSP very difficult because of the overlap and duplication that exists. Armstrong[20] suggests that approximately 90% of the separations performed by the 50 or so currently available CSP's could be handled by as few as six or seven of them. The use of multimodal chiral CSPs can be viewed as a possible solution for this dilemma, which function as two or three columns depending on the separation conditions. Thus multimodal CSPs can resolve different types of enantiomers through utilizing different chromatographic modes.

A CSP must meet several criteria before it can qualify as multimodal[20]. Stationary phase stability in most solvents and mixtures is an obvious necessity, but is not sufficient by itself. There must also exist significant selectivity and mechanistic differences in each of its modes. These two criteria must be exhibited by CSPs proposing to be multimodal. In addition, the selection of

modes and optimization of separations should proceed in a systematic and logical fashion.

II.A.4. Chromatographic Expert Systems

There have been a number of expert systems covering different areas of chromatography reported in the literature. Many of them revolved around the area of column and mobile phase selection in HPLC. Refer to section II.B.6 for several non-chromatographic expert systems developed in the field of chemistry.

These chromatographic expert systems had the benefit of utilizing either mature theories and mathematical equations governing the many interactions that occur within a chromatographic separation, or well studied empirical relationships[22]. One commonly used equation[23] relates retention as a function of mobile phase composition (1):

$$\log k' = \log k_w - S\Phi \qquad (1)$$

where k′ is the capacity factor of a given solute for a mobile phase of composition Φ, which is the volume fraction of the strong solvent in the mobile phase, and both k_w and S are characteristic constants of the strong solvent and solute. Other relationships have been used as criteria for selectivity optimization[24]. Further discussion of these expert systems is warranted.

ECAT, an acronym for Expert Chromatographic Assistance Team, is under development at Varian Associates[25]. It was the first expert system in the field of chromatography, announced in 1984[26]. Dr. Karnicky assisted in the development of an expert system to design, analyze, optimize, and trouble-shoot an HPLC separation method[27]. The system can specify the column and mobile phase suitable for reverse phase chromatography of common classes of organic molecules, reverse phase, proteins and peptides. A limited set of specialty classes of molecules best treated by straight phase chromatography can also be specified. Some column diagnosis can be performed based on factors such as efficiency, asymmetry, retention time, selectivity and operating pressure.

Williams, Excoffier and Abbot[28,29] have described recent work on the project. ECAT is currently comprised of four different modules with a total of almost one thousand rules and facts. The CMP module (Column and Mobile Phase) bases its recommendations for a prospective method on information describing fundamental properties of the of the sample being analyzed input by the user. Recommendations include a mode of operation, a column, a mobile phase, and a detector. MOP (Method OPtimization) is used to specify a gradient to use for the mobile phase, a task usually determined experimentally. Sample pretreatment prior to LC

analysis, is typically required for real samples to remove the analytes from the matrix in which they exist. User selection of such a method is assisted with the SPR (Sample PReparation) module.

The Varian Inference System (VIS) combines the above modules with a molecular property database module to make up the ECAT system. ECAT is written in Common LISP and runs in both the VAX and PC environments. Future work will focus on the development of a separate module for detector selection, and as well as a master controller for all of the modules.

There are currently a large number of scientists in Europe working on ESCA[30], a project on the "Application of Expert Systems in Industrial Chemical Analysis." This work is supported by the EEC (Eastern European Commission) through ESPRIT (European Strategic Program for Research and Development in Information Technology) Project 1570. The direction of research under project ESCA is in demonstrating the applicability of expert systems to HPLC with an emphasis on pharmaceutical analysis, and to compare tools and methods for doing so. ESCA schematically describes the HPLC method development process as a series of 4 domains (see Figure 4[30]): Domain 1 provides initial chromatographic conditions, Domain 2 performs chromatographic optimization, Domain 3 addresses method validation, and Domain 4

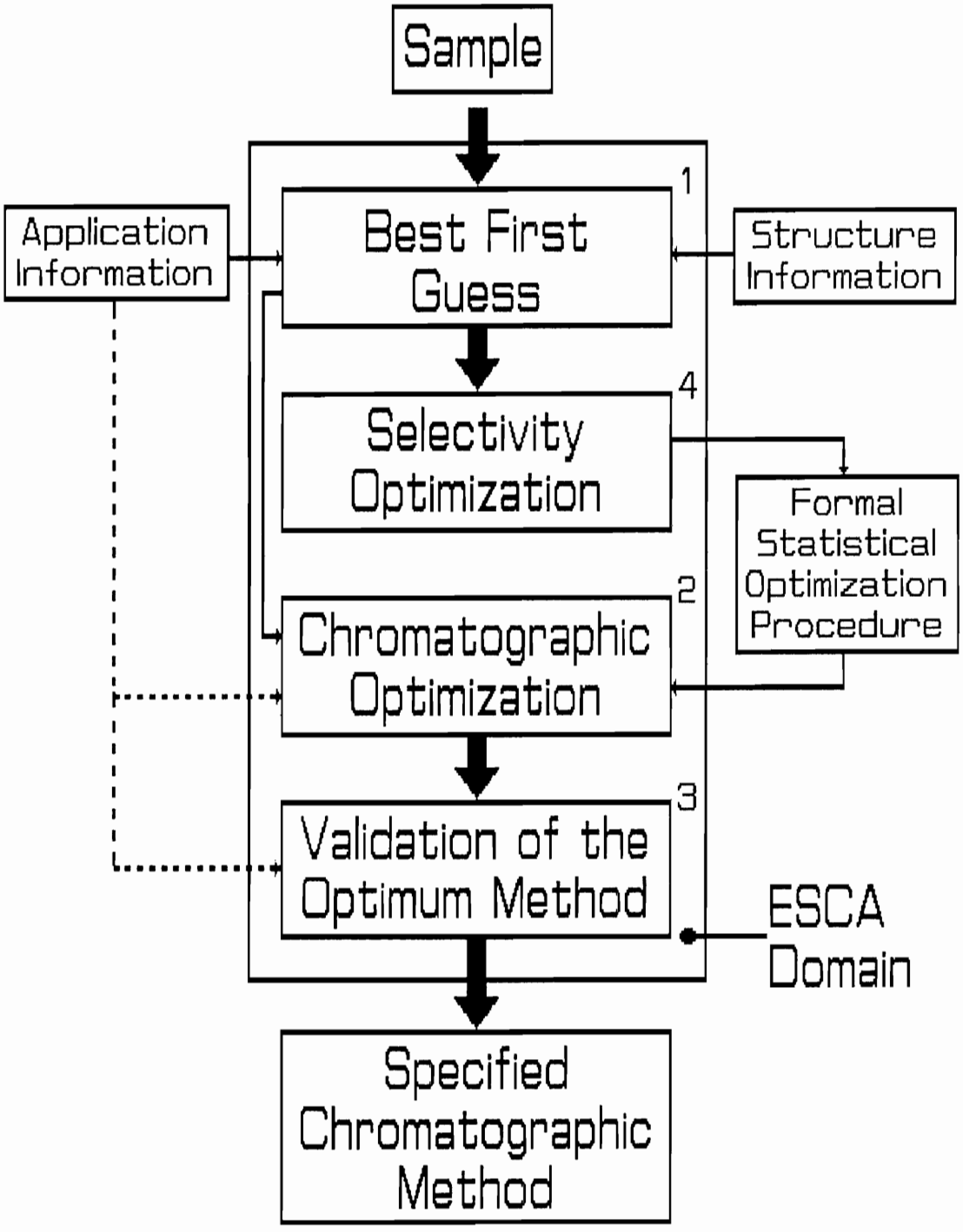


Figure 4: ESCA Project Domain

encompasses selectivity optimization. Groups of researchers are addressing each of these domains.

Massart and Musch from the University of Brussels designed an expert system that helps the analyst in making decisions about method selection and development specifically in pharmaceutical and biomedical analysis[31,32]. This initial work was restricted to HPLC, GLC, and UV spectroscopy. Researchers completed the development of rules for the selection of the detection method in HPLC, as well as a feasibility study concerning the use of expert systems for the development of procedures in pharmaceutical analysis. It was concluded that it is indeed feasible to develop expert systems for the selection of analytical methods and procedures.

Progress in this project has recently been reported, and is now part of project ESCA[33]. The entire project is made up of nine modules organized into several groups of stand-alone systems: a first guess group (ESCA domain 1), a retention optimization group, and a selectivity optimization group (ESCA domain 4).

Three stand alone expert systems make up the first group. LABEL is a first guess expert system for label claim analysis of pharmaceutical formulations[34,35]. DASH is a first guess expert system for the purity control of drugs which act on the central nervous system[36]. LIT allows

methods from the literature to be introduced. An experiment is performed with the set of initial chromatographic conditions suggested by one of the expert systems from the first guess group.

The group of retention optimization expert system modules are called if the capacity factor (k') is outside of the desired range of values. Based on the first guess results, LABEL', DASH', and LIT' determine new experimental conditions. This group of multi-level systems can be utilized in a recursive fashion up to three times or until a satisfactory retention is obtained.

If after obtaining satisfactory retention of the compounds of interest, it may be necessary to enhance the peak resolution of the chromatogram to optimize separation. SLOPES is a selectivity optimization expert system consisting of three modules: the first decides on which parameters to optimize and their boundaries, the second determines the type of experimental design to utilize, and the last selects most suitable optimization criterion. A stand alone expert system for the third module, the selection of criteria for selectivity optimization, has been developed[37].

The software used by the project was driven by their hardware selection: an IBM PS2/80. The commercial expert system building tool KES.PS (Knowledge Engineering System)

was chosen[38]. The integration of the above modules, all written in KES, was achieved through the development of SES, the supervisor expert system. SES acts as a scheduler, and decides under what circumstances each of the individual modules are required to obtain a solution. Each of the modules was embedded in a C-language program, called at the appropriate time by SES. Interaction between modules occurs via saved files. Work on this project continues under funding provided by ESPRIT Project 1570 under ESCA.

Buydens, van Leeuwen, Mulholland, Vandeginste, and Kateman are investigating a method's development expert system (ESCA Domain 3)[39]. This system begins with an HPLC method that has been selected and optimized in the other expert systems of ESCA and validates it, testing such performance aspects as accuracy, precision, sensitivity, selectivity, and limitations.

The estimation of precision is the fundamental property used by this expert system in the testing of an HPLC method. Because of the more or less complex statistics required to establish the precision of a method, most analysts consider it a tedious task. The expert system addresses this problem by proposing the experiments to be performed, applying the appropriate statistical tests, interpreting the results, and suggesting changes to the method if the test fails.

Validation resulted in a success rate of approximately 85%, based on presenting the system with 11 real cases.

Computer simulation is another way to assist chromatographers in performing separations. In fact, commercial systems consisting of hardware and software are available[40]. Such systems, rather than being called expert systems, are often described as optimization systems that make use of computer simulation to arrive at acceptable values for separation variables such as type of solvents in the mobile phase, pH, and concentration. However, virtually all of these do not suggest a column to use, but optimize the separation for the column chosen by the chromatographer.

Drylab[41] is a series of software tools that utilizes computer simulation to predict optimal separation conditions based on the outcome of several initial experiments. After these results are entered, the software performs simulations on how mobile phase composition, gradient slope, and column dimensions might effect the separation. Optimum separation zones are identified using the resulting resolution maps. The suggested separation can then be performed, yielding a savings in time and supplies. While these packages rely mostly on varying solvent strength to affect changes in band spacing, i.e. solvent strength optimization, they can also be used for optimization of pH, ternary solvent mixtures and temperature. Drylab relies on empirical relationships

relating such parameters as retention to mobile phase composition, and plate number as a function of experimental conditions[42,43].

The field of chiral chromatography has not yet developed to a similar point. Although all of the equations mentioned above are still valid, they are not sufficient to describe the interactions that are necessary to separate isomers. While relationships are now beginning to be developed (see the next section below), universal equations governing such separations are only now in the early stages of investigation.

II.A.5. Predicting Chiral Separations

In an effort to understand both the mechanism and theory behind chiral separations, a number of efforts involving such techniques as symmetry measures, modeling, and energy calculations have been performed. These studies range from theoretical investigations into the symmetry around the chiral carbon, to approaches based solely on experimental results. The effect of such studies will be to lead researchers to a more complete understanding of what causes chiral separations. Several of these studies warrant further discussion.

Berthod, Chang, and Armstrong[44] have recently published a work that uses free energy calculations to

predict separation on cyclodextrin CSP. Their work represents the capacity factor (k') in terms of a molecular free energy of adsorption, and uses it to define enantioselectivity (α) as a ratio of the capacity factors of the two enantiomers. All molecular free energies are equal for the enantiomers (and hence cancel) except those responsible for enantioselectivity. The resulting simplified equation shows a direct relationship between α and the molecular free energy of the chiral interaction for the two enantiomers, as first defined by Pirkle and Pochapsky[16]. This is used as a measure of the degree of chiral recognition between a given CSP and the two enantiomers of interest.

In order to calculate this molecular free energy of chiral interaction, two assumptions were made. The first was that the free energy was the sum of four terms, each related to one of the four substituents attached around the chiral center. The second is that the chiral interaction contribution of any substituent is independent of the others. Taking these into account, once values for substituents were known, the separation factor (α) of enantiomers could be calculated for the column used in the study by summing the contributions from each substituent.

A statistical fitting method was used to arrive at the individual substituent contributions. The equation relating

the separation factor to the molecular free energies reduces to an equation with three unknowns, each one representing one substituent (one of the substituents is always Hydrogen, which was used as the reference and defined to be zero). Data from studies conducted on 121 chiral molecules with one cyclodextrin CSP (both **R** and **S** forms) to generate a substituent energy list for 81 different substituents was entered into a computer spreadsheet, using one equation (with three unknowns) for each molecule studied. The problem reduced to solving 121 equations, each with three unknowns, and then minimizing the absolute value of the difference between the calculated and experimentally determined selectivity.

The enantioselectivity of a particular compound is calculated by combining the energy contributions of each of the four substituents connected to the chiral center. Using this methodology, based only on experimental results, one can empirically predict a priori whether or not a CSP will separate a compound (but not the elution order) and estimate its separation factor (α). Provided that a sufficient number of molecules are used to generate the energy lists for any given CSP, this approach can theoretically be used for most CSPs.

Lipkowitz and Baker[45] developed a theory that enables them to determine the relative retention order and

separability factor, α, for several commercially available Pirkle type CSPs. This work is based on calculating binding energies of optical isomers to chiral surfaces[46]. The transient species that result in enantioselectivity are modeled as weakly bonded diastereomeric complexes, and their relative free energies calculated using the MNDO method as employed in MOPAC. The work also provides a convincing argument as to whether one could base predictions at all on such small differences of internal energies[47].

Their model enables them to begin to understand the contributions of the CSP to both enantioselectivity and analyte binding. They found no indication that for their system, the two enantiomers bind at different sites on the CSP. Evidence also suggests that the molecular fragment containing the chiral center in the CSP is not the dominant fragment in chiral recognition. In fact, for the systems studied, their role may be reduced to providing a steric barrier, or even to controlling the conformation of other parts of the molecule. This is consistent with the view purported by Mislow[48] that not just the asymmetric center, but the entire molecule is chiral.

Topiol, Sabio, Moroz, and Caldwell[49] performed computational studies of the interactions between the chiral molecule and CSP based on molecular mechanics and semiempirical and ab initio quantum mechanical calculations.

Results suggest that small, through-space field effects are responsible for enantiomer differentiation, rather than the classical Pirkle model discussed above. They suggest that the π functional groups do not appear to provide any selectivity between the two enantiomers studied since the π groups contribute equally to the elution of both enantiomers.

Buda and Mislow[50] have taken a drastically different approach to quantifying chirality. They propose that chirality is a property that is independent of its physical and chemical manifestations. The absence of improper rotations in the symmetry group of the system is the ONLY requirement for a system to be chiral. They point out that Kelvin[51], in his original definition, formulated the concept of chirality as an abstract property of geometric objects:

> "I call any geometrical figure, or group of points, chiral... if its image in a plane mirror, ideally realized, cannot be brought to coincide with itself".

Thus, chirality is reduced to the geometric relationship between points in space (atoms), i.e. a function of an objects shape. Critical discussion and reviews of this area are available[9] (referring especially to reference 10 therein).

Their chirality measure is based on Hausdorff's concept of distances between sets. This is a natural choice since sets of atomic coordinates in 3-dimensional Euclidean space are often used to represent molecular structures. Although enantiomers are all based roughly on a tetrahedron shape around the chiral carbon, each molecule's unique combination of atoms, bonds, and intramolecular interactions will vary its exact geometry. Building on this definition, it becomes possible to speak of a system being more or less chiral or achiral. Their work, based solely on geometric shape, is an attempt to quantify these variations. Although this work has thus far been limited to geometric figures and to rigid models of molecules, their goal is that eventually the technique will prove useful in correlating molecular shape with properties arising from molecular chirality.

II.B. Chemistry-Based Expert System Shell

An expert system shell is a collection of tools that assists in the development of expert systems, a toolbox if you will. To assist in the development of expert systems in the scientific arena, an expert system shell was developed that understands a number of fundamental principles of chemistry. CHESS, first reported in a preliminary form in 1988[52], has now been implemented essentially as originally envisioned. It entails "teaching" an expert system shell

principles of chemistry such as atoms, bonds, functional groups, molecular properties, and molecular structures. What follows is some background information on what led to the development of CHESS and how it can be used to develop expert systems that can reason based on chemical structures and properties.

II.B.1. Molecular Representation

The recognition and manipulation of molecular structures is imperative to any expert system designed for the chemical arena. This topic divides itself into two areas: what the internal molecular representation format will be, and how to input the molecule information into the system.

There are a number of standard methods for representing molecular structures for input into computer programs. The Wiswesser Line Notation[53] is perhaps the most widely used symbolic string line notation for chemical structures. The output generated is a deterministic linear string of symbols that can be easily entered into computers. However, the myriad of rules needed in order to generate complex structures make it difficult to utilize.

SMILES[54] (Simplified Molecular Input Line System) was developed combining elements of graph theory to produce a very small and natural grammar that allows rigorous

structure specification. Its use of atomic symbols (instead of a more compact representation) and structural symbols (for bonds and branches) is aimed at yielding a representation that is readable to the human eye rather than yielding an optimized storage representation. SMILES concise form denotes a two-dimensional valence-oriented graph of a molecular structure that is ideal for computational applications. The three-dimensional spacial orientation of atoms within a molecule is not addressed with this system.

Strings of character symbols representing chemical structures are easily recognized as input by computer programs. Regardless of which of the line notations is used, scientists do not think of structures as lines of symbols. A graphical approach to molecular structures is preferable.

There are a number of graphical tools that can be used to produce chemical structural information. Common to most of them is the standard connectivity table approach originally developed by Wipke[55], and incorporated into MDL's[15] MACCS, ChemBase, ChemText, ChemTalk, and ISIS products. MDL refers to the connectivity table as a MOL file. In developing CHIRULE, ChemTalk was used to interact with MACCS running on a VAX mainframe computer to generate the MOL files (Refer to Chapter VII for a further discussion

of the computing facilities used in this project). CHESS uses both MOL files as well as SMILES strings. The SMILES strings are used by SPARC to calculate various molecular properties based on molecular structure. The MOL files are used by CHESS for all functional group searches (see Chapter III.A for a complete description of the MOL file format).

Each of these formats have their own advantages and disadvantages, which will not be fully expounded upon here. Synthesis, quantum chemistry, and molecular modeling systems all place different demands on a representation format. As it is not expected that one particular format will ever meet the wants and needs of all developers within this particularly diverse area, different molecular representation formats will continue to be utilized.

Perhaps a palatable compromise would be to settle on a standardized format for the exchange of information between programs. The SMD[56] (Standard Molecular Data) format is one such standard that has developed momentum in this area, and merits noting. Its exhaustive definitions and extendibility provides a vehicle for representing chemical data, and is envisioned as an exchange medium[57]. In fact, a compatible format with a more limited scope has already been implemented. The JCAMP[57,58] (Joint Committee on Atomic and Molecular Physics) format emphasizes stereochemistry and other spectroscopically relevant features.

II.B.2. Functional Group Recognition

Computer searching for functional groups within molecules is a very intriguing problem. To date, many efforts have focused on treating functional groups as a part of the overall structure, or as a substructure. Searching for these substructures is accomplished by looking through a molecule database for all cases that contain a certain collection of appropriately connected atoms. However, the results of these searches can be less than ideal. Let's take two molecules as examples: Phenylglycine and Phenylalanine (see Figure 5). These molecules have a part of their structures in common. This common sub-structure is known as Glycine. Doing a substructure search on Glycine utilizing MACCS does in fact yield both of our target molecules, along with over 900 other molecules[59]!

This technique relies on being able to define a functional group and what one might call its major atom, the atom by which it is connected to the rest of the molecule. Esack and Bersohn[60] devised a program to automatically recognize functional groups based on querying the canonical connection table for all functional groups in the library. Their work differentiates almost 50 functional groups and their representative central atom (see also Chapter III.B).

The system, written in assembly language on an IBM main frame, is targeted to reducing search times for computer

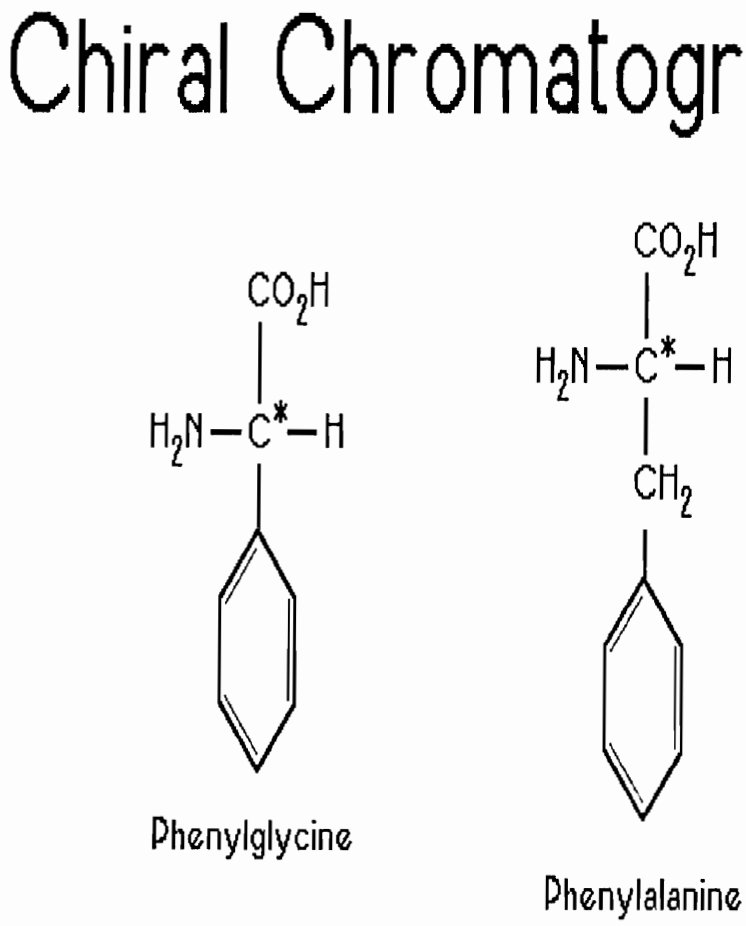


Figure 5: Similarity Searching Targets

analysis of mass spectra or computing synthetic pathways. They suggest that many current search procedures involve developing a list of functional groups and repeatedly querying the molecular representation as to each one's presence. Even though screening techniques can be applied, it is time consuming and inefficient due to the large number of negative results.

The representation of functional groups is explicitly built into their FXNLIST table, and hence there is no need to "search" for them. This FXNLIST, built from their own unique canonical connection table format, could also be constructed from the more popular MOL file format. It is arguable whether the speed gained by the use of the FXNLIST is offset by the need to generate and store the specialized structures (the atom machine formula and group machine formula for example[61]) required to create the FXNLIST. The use of MOL files, with their implicit representation of functional groups, seems to be preferred to Esack and Bersohn's methodology for representing functional groups explicitly in a separate structure that is in addition to the molecular representation. These separate structures unnecessarily complicate the development of functional group searches. The use of a more "english-like" syntax is made possible by using MOL files, and is imperative if other researchers are to be able to build on the work already

accomplished. Chapter III discusses this aspect in great detail.

Another work by Armstrong and Hibbert[62] mentions, but does not expound upon, the advantages of using the language Prolog in the matching of chemical structures. In their work, a chemical structure is converted into a symbolic name, or s-name, that can be represented as a Prolog data structure. This data structure is based on the topology of the structure, and consists of a list of all bonds found within the molecule along with the respective atoms on each end of the bond. The s-name can then be converted into another structure, the generic name or g-name, that serves as a key by which the s-names can be ordered, making the storing and efficient accessing of chemicals in a large chemical data base possible. While the s-name is unambiguous, it is not unique. However, one can differentiate between two different encodings of the same structure by deciding if the two s-names are isomorphic.

There are several short comings to this approach. Although these workers did capitalize on the searching capabilities and automatic backtracking of Prolog, they failed to develop why this is so important to such areas as functional group recognition (refer to chapter VI for a complete discussion). They also did not expand it to higher order bonds (double or triple) nor to ring structures

(although they do propose a method by which both could be done that is reminiscent of the connectivity table and SMILES string approach). This precluded them from addressing the topic of sub-structure searching.

II.B.3. Property Estimation

The capability to calculate properties of a molecule based solely on its structure is an area that is experiencing renewed interest[63]. The EPA became interested since it provided a mechanism to estimate toxicity, reactivity, and transport parameters for thousands of industrial compounds without requiring the expense and time of traditional laboratory testing. The use of such Quantitative Structure Activity Relationships (QSAR) could increase considerably if current limitations can be overcome. Expert systems that base predictions on fundamental chemical structure theory and QSAR methods that use readily available spectroscopic data are techniques that are being investigated.

A joint project between the U.S. EPA Environmental Research Laboratory (Duluth, Minnesota) and Montana State University (Bozeman, Montana)[64] has resulted in a system that estimates molecular properties, environmental fate, and toxicity for organic chemical compounds using models based solely upon chemical structure. The Montana/EPA system will

calculate the following properties: molecular weight, LogP, heat of vaporization, parachor, melting and boiling point, pKa, molecular refraction, water solubility, molar volume, vapor pressure, and surface area. It runs on a VAX mainframe environment under VMS. Recent work with the Montana/EPA system involves its incorporation into the ASTER[65] (ASsessment Tools for the Evaluation of Risk) system.

CAChe Scientific has another collection of programs that performs property estimation[66]. The program, called CAChe, is a collection of computational chemistry packages that are joined together under a common graphical user interface. Its strength lies in its visualization techniques[67]. Parameters such as electrophilicity, nucleophilicity, solubility, activation energies, and IR/UV/Vis spectra can be predicted and visualized.

SPARC (SPARC Performs Automated Reasoning in Chemistry) is an expert system that estimates chemical and physical reactivity[63]. Based on these estimates, SPARC calculates various properties based solely on a molecule's chemical structure utilizing the reasoning process that a scientist might undertake in reactivity analysis[68]. It is a joint project between the EPA and the University of Georgia.

The SPARC approach goes beyond more traditional statistical QSAR techniques that rely on regression

coefficients, by calculating properties based on physical principles. Extensions of linear free-energy theory and perturbation theory are critical to SPARC's calculations. Current properties include molecular polarizability, hydrogen bonding parameters, molecular weight, molecular volume, solubility, vapor pressure, heat of vaporization, distribution coefficients, boiling points, index of refractions, and Henry's constants. Work continues on including a pKa estimator and an absorption spectra generator.

Attributes used in the selecting a package to add molecular property estimation capabilities to CHESS included ease of incorporation, properties provided, and availability. SPARC excelled in all of these areas. A batch mode allows CHESS to generate all the information required by SPARC to gain access to the property estimation capabilities. This greatly simplified SPARC's integration into CHESS. SPARC was the only package to provide a broad range of properties that were appropriate in modelling the chemical environment around the chiral carbon, (molecular polarizability, hydrogen bonding parameters, and molecular volume were chosen). Dr. Karickhoff and Dr. Carreira, co-developers, became interested in the research and agreed to make a pre-beta version of SPARC available for use in this work.

II.B.4. Similarity Searching

The explosion in both the number and size of commercial and proprietary chemical databases has placed increased pressure on methods used to access that information. Any collection of molecules can be thought of as being present in some sort of a "chemical information space", with each molecule being a point in that space. In querying that space, the user makes an attempt to "mine" all relevant molecules[69]. The query, in effect, uses a molecular descriptor to map those molecules into another space, a "molecular similarity space." This space is ordered according to the molecular description, allowing the user to more easily extract the information that has "floated" to the top[70].

There are many different criteria that can be utilized in defining the molecular descriptor. Historically, the first methods used in searching were based on an exact match with the target molecule. Although this method did provide a technique for retrieving stored information about a specific molecule, it was not effective at mining the chemical information space.

In an effort to expand searching capabilities "wild cards" could be substituted for portions of the molecule, allowing a search for sub-structures related to the target molecule. Standard substructure searching allows the user

to specify a string of substructures (i.e. functional groups) in a Horn clause fashion (i.e. WWW and XXX and {YYY OR ZZZ}). The search will elicit all molecules from the library that successfully trigger the Horn clause. Often times it is not the physical structure of a molecule that is important, but something else.

Similarity Searching is a more recent technique for finding "similar" matches with a given molecule by concentrating on some physical and/or chemical reactivity attributes of a sub-unit of the molecule. These sub-units can range from specific atoms, to functional groups, and even to fragments of molecules. This concept of similarity, based on properties of the molecule, is not limited to any specific subset of functional groups. It does require specifying a set of descriptors to be used in defining similarity.

Bawden points out that there are four main types of descriptors that are used. The first two have found applicability in similarity assessment[71]. Distance coefficients are a measure of the distance between two items in an n-dimensional space. The Euclidean distance coefficient is the best-known example of this type, which is widely used because of its ready geometric interpretation. Association coefficients are often used when handling binary items, such as the presence or absence of a structural

feature. Correlation coefficients are not generally used in similarity searching, although they have found wide use in general statistical analysis. This is due to the fact that the similarity between the parent objects is not necessarily implied simply by a linear relationship between sets of variables representing them. Probabilistic coefficients, which take into account the distribution of variables over a set of items, have been found to perform poorly in studies on molecular similarity[71].

The concept of a molecular similarity space can be augmented with scalar properties, such as molecular properties (dipole moments), chemical properties (boiling points), or even biological properties (the binding coefficient of a drug)[71]. This is why similarity searching is playing an ever increasing role in querying chemical databases. More details may be found in a recent Workshop on Similarity in Organic Chemistry[72].

Substructure searching uses functional groups and wild cards, in comparison to a similarity search which focuses on sub-units based on some physical and/or chemical reactivity attributes. Sub-structure searching is somewhat more rigid, where similarity searching is more flexible. In the CHIRULE system, similarity can be defined up front, which relieves the user of the responsibility for implicitly building the ideas of similarity into search requests. For example, in

example, in searching for structures similar to a particular molecule, substructure searching would require the user come up with a subset of functional groups that defines all the types of target molecules that are of interest. CHIRULE's similarity search only requires the "ideal" or "target" molecule[71], thus relieving the user from the task of defining similarity.

II.B.5. Automated Knowledge Acquisition

The key to introducing expert systems to scientists is providing a tool by which they can experiment with it on their own. The problem automated knowledge acquisition has to solve is finding a technique to regularize knowledge; that is, how to fit an expert's knowledge into some sort of generic template. Perhaps a more appropriate way to state this is how to come up with a conceptual model that can adequately represent the expert's knowledge. There has been considerable effort invested into approaches that can produce such models. This section documents the investigation of several that appear to show promise. Complete discussions of the entire area of knowledge acquisition, both automated and manual, are available in the literature[73].

Knowledge acquisition, or the gathering of information for inclusion into a knowledge base, can be a very labor

intensive process if it is done manually. There are several obvious ways knowledge can be obtained in a form usable in an expert system. Being told about an area and learning by example are the two easiest of them. More difficult methods include observation, discovery, experimentation, and reasoning from knowledge representations.

Entering information into the knowledge base using a text editor is simple for a scientist. It has the advantage of not requiring complex data entry routines. However, there is concern about data integrity. This method is satisfactory for smaller systems, and simplifies their design. However, it would be too time consuming to use for larger systems.

The most prevalent method of gathering knowledge is by exhaustive interviews conducted by a knowledge engineer who has extensive training in knowledge representation and expert systems. In most cases, the knowledge engineer will sit down with the expert and, in several meetings over a number of weeks, ask questions that extract the knowledge from the expert. The knowledge engineer takes detailed notes of their conversations, and later transcribes them into the knowledge structures that make up the expert system. This is a very labor intensive method, and is further complicated by the personalities of the people

involved, their moods on a given day, and other factors having nothing to do with the knowledge itself.

There are several methods that can be used to speed knowledge acquisition. One of them is to have the expert system develop rules from examples. This method, referred to as induction, requires a training set of examples from which attributes are established. These attributes allow comparisons to be made among the examples found in the training set. The rules are derived from these comparisons. Induction enables the transplanting of chunks of decision making knowledge from an expert onto a computer. Professor Ross Quinlan of New South Wales Institute of Science and Technology[74] developed a machine procedure that allows such skill transplants. His Pascal-coded program is called ID3, which stands for Iterative Dichotomiser 3. The following can be concluded from his work:

1) The time required for such a program to generate machine-executable solutions for complex decision problems is significantly less than if it were to be done by conventional hand coding.
2) The resulting solutions are also very efficient.
3) It is imperative that a decision be made previously on whether only efficiency of the machine executable code is required, or if the resulting

knowledge base must also be understandable by human inspection.

While the resulting system is very dependent on the training set used, this technique is finding use in chemical data systems[75]. If the data set contains sufficient information, results will be satisfactory. If the training set is in any way incomplete, or not sufficiently specific, however, these defects will be evident in the final system. In addition, if an understandable knowledge base is a desired result, treating a problem as one large super-problem with a single associated file of examples must be avoided. This can be accomplished using Structured Induction[74] which involves separating the problem with its file of examples into sub-problems and even into sub-sub-problems if necessary.

A second method useful in knowledge acquisition is Personal Construct Theory. It is based upon a clinical psychotherapeutic interviewing method originally developed by George Kelly[76]. Kelly believed individuals seek to predict and control the events in their lives by forming theories, testing hypotheses and weighing experimental evidence. This led to Kelly's theory of a "personal scientist". Based on this theory, he developed the Repertory Grid Test for eliciting role models. The grid used to develop his model is similar in form to a

spreadsheet. Clients are asked to list, compare and rate role models in their lives (referred to as elements) to derive and analyze character traits. These elements (such as a father figure, wife, or a close friend) are then placed along the top of a grid. Taking a combination of three of these elements, the client is asked to identify a characteristic trait that distinguishes two of the elements from the third, along with an opposite of that trait. Representing an internal bipolar scaled construct, the trait is placed on the left side of the grid, and its opposite on the right. Traits such as friendly/unfriendly, rough/gentle, and loving/mean were among the ones commonly used in the original theory. The grid is then filled in with information in the form of an "X", showing that the element has that particular trait. A blank means the element did not have that trait (or has the opposite of the trait).

Gaines and Shaw[77] viewed the grid rows and columns as assignments of truth-values to logical predicates, suggesting that the grid is simply a matrix of truth values, and that the one side is just the logical negation of the other. The original bi-polar scale of either "X" or a blank can be expanded to a scale of 1 to 5. For one of the traits mentioned above, a grid value of 5 would mean "friendly" and 1 would mean "unfriendly". Other scales, such as 1 to 7,

have been used. In addition, a weighing factor can be added to the traits themselves, allowing some traits to be more important than others. The authors cited used fuzzy set theory to derive implications from the grid, and based on the multivalued logics and the trait weights, calculated an implication strength.

The connection of these techniques to automated knowledge acquisition arises from the idea that rules can be obtained from the grid utilizing techniques such as methods based on topology. Rules take the general form of stating that a particular element is likely to be the correct answer, with a confidence factor, if a trait has a certain value. The implication strengths mentioned above translate into the confidence factors for the rules.

One of the real advantages of the technique arises from its ability to test the grid for both completeness and ambiguities using techniques such as correlation and cluster analyses. If in performing the analyses it is determined that two of the elements are nearly identical, another trait should be elicited to further distinguish between them. If it is determined that they are in fact identical, combining them will result in making the information contained within the grid complete. For example, it was determined that the grid may represent both your wife and your close friend as possessing the same traits. If you married your close

friend, it may be appropriate to simply combine the two elements (wife and close friend) by deleting one of them. However, if the two are not identical, another trait (such as logical/illogical) should be used to further distinguish between them.

The traits can be analyzed in a similar fashion. If the trait friendly/unfriendly was found to be similar to loving/mean, one could be removed to eliminate this ambiguity. However, cluster analysis can also determine if one trait is the exact opposite of another. For example, loving/mean might be the exact opposite of rough/gentle, in which case one of them could also be removed. This can occur because of the bi-polar nature of the traits.

Further research into utilizing Kelly's[76] Personal Construct Theory to aid the translation of knowledge into an appropriate conceptual model suggests that this method alone will not be sufficient as a universal knowledge acquisition tool. Personal constructs lack the capacity to represent problems that cannot be broken down into a decision tree structure. However, it will still play an important role in a final solution, since it does handle classification problems quite adequately.

There are several commercially available tools that are based on Kelly's work on Personal Construct Theory. Auto-Intelligence from Intelligence-Ware[78], and Aquinas from

Boeing Computer Services[79] are two programs that aid in building expert systems employing Kelly's theory. Another commercial product is Knowledge Garden's KnowledgeMaker, which claims the ability to produce rules from either text or LOTUS 1-2-3 files.

Auto-Intelligence is promoted as the first commercial interactive knowledge acquisition product for the PC environment developed to capture knowledge for structured selection or heuristic classification problems by producing rules from examples input by the user. This interaction enables the program to identify important constructs and criteria, often unknown to the expert, used in the decision making process. Interrelationships between criteria and their importance are also analyzed, helping the expert to clarify their own thoughts as to their relative merits. Next Auto-Intelligence assists in the examination of both data and examples for redundancy and inconsistencies. Following corrections, knowledge is generated using induction, and rules can be derived to form an expert system.

There are many strengths that AutoIntelligence possesses. The ability to represent both traits (which take on relative values: ex. color) and attributes (which take on specific values: ex. melting point), and the ability to combine these values when deriving confidence values are

examples of such strengths. However, there are several issues that have been left unanswered. What mechanisms are used to detect redundancies? How are certainty factors handled? Why is the user limited to only seven criteria (or elements) at a time? In developing CHESS, questions such as these have been thoroughly researched to determine investigators how best to reach a solution. For example, correlation and cluster analysis have proven effective at insuring the completeness of even very large grids. This and other topics are further discussed in Chapter IV.

Boeing's Aquinas software is the result of continuing work on the Expertise Transfer System (ETS) over the last three years. Aquinas's strengths lie in helping users further define the problem they are trying to represent, and determining the feasibility of the project. Due to its limitations in knowledge representation and reasoning, ETS was often abandoned during the later stages of the knowledge acquisition process. Aquinas is an expanded version of ETS that is aimed specifically at overcoming some of these difficulties.

Both Aquinas and ETS were developed at Boeing, are very powerful, and represent many hours of development work. They are heavily graphics and window oriented, and run on larger computer systems. Reviews of the capabilities of both ETS[80] and Aquinas[81] have been published.

Several other approaches have shown some promise. Davis[82], in his work on TEIRESIAS, used what he termed meta-level knowledge as a way to develop a system that has the capability to use its knowledge directly and the competence to examine it, abstract it, and direct its application. Meta-level knowledge can be thought of as "knowing what you know." A system that accomplishes this has the ability to use its knowledge directly by examining itself to discover where inadequacies exist.

Roach[83], in his work on robot planning, describes a "Know/No-Know" knowledge representation that provides systems with the ability to reason about their own limitations and make adjustments accordingly. It addresses the other side of this issue, that of knowing what you do not know. Knowing whether a fact is thought to be true, thought to be false, or that its truth is uncertain or unknown can help the system in planning and executing its assigned tasks. Another approach that is being explored is J. Wade's[84] "person object spectrum". Some initial research into applying Wade's work towards an automated design consultant has been conducted by Cathey[85].

System for Interactive Transfer of Expertise, or SITE[86], combines methodology based on cognitive studies of human thinking with systems dynamic modelling (SDM) in an effort to extract knowledge from an expert. SDM is based on

the premise that the dynamic behavior of a system is a result of the presence of feedback loops within the problem structure. The goal is to make obvious the effects of changes in variables to the overall behavior of the system. This way beneficial variations can be promoted while inhibiting the undesirable variations. The model that results is referred to as an influence diagram. This consists of nodes representing the system variables, and arcs connecting them representing causal links. This influence can then be converted into a series of rules. There are a wide range of applications using this methodology, such as inventory tracking and modelling the production sector of the economy.

Any final solution to the automated knowledge acquisition problem would consist of the combination of several different methodologies, each suited to solving a different type of knowledge acquisition problem. Personal Construct Theory adequately represents classification domains, and hence would be one of the tools incorporated into the system. However, when this tool chest of techniques was assembled, there would need to be a way to choose which tool was most appropriate to the particular knowledge that the user wanted to record. This requires classifying and matching the knowledge that the user possesses with one of the available tools. Assisting the

user in choosing a knowledge acquisition tool is another role Personal Construct Theory could play in such a system.

Using Personal Construct Theory as the tool chest manager provides a mechanism to access the existing knowledge acquisition techniques. In addition, it allows new commercial knowledge acquisition tools to be incorporated into the system. Desirable commercial systems are bound to be developed in the coming years as the technology matures, and the ability to incorporate them into an existing system is an essential part of the expert system shell concept.

II.B.6. Expert Systems

Expert systems, a part of artificial intelligence (AI), have been developing over the past twenty years. In the early 1970's, many people thought the field of AI was going to be able to solve such complex problems in computer science as speech recognition and other natural language topics, as well as the computer representation of human knowledge. Many promises were made which added to the "hype" that has surrounded this area. When these promises could not be kept the public at large lost confidence in AI. Today expert systems are also being subjected to much of this same hype. In order to better understand the

utility of expert systems, let us examine what exactly an expert system is.[87]

The words "artificial intelligence" themselves suggest a goal of human equivalence achieved by inhuman means. Behind them lies the fundamental paradox of creation versus analysis: if computers can only analyze but not create, how can they be made intelligent? Initially it was hoped that the door of intelligence could be unlocked by a relatively small number of general, yet powerful mechanisms. These mechanisms could then be used to create a computer model of the symbolic processing readily done by the human mind. However, computers used as symbolic processors fell short of providing a basis for solving complex practical problems readily solved by humans, despite advances in searching, manipulation, and representation of data in symbolic data structures. Out of this shortcoming grew an understanding of the importance of knowledge representation. Goldstein and Papert[88] identified this importance and realized the effects it would have on artificial intelligence research:

> "Today there has been a shift in paradigm. The fundamental problem of understanding intelligence is not the identification of a few powerful techniques, but rather the question of how to represent large amounts of knowledge in a fashion that permits their effective use and

> interaction.... The current point of view is that the problem solver (whether man or machine) must know explicitly how to use its knowledge with general techniques supplemented by domain-specific pragmatic know how. Thus, we see AI as having shifted from a POWER-BASED strategy for achieving intelligence to a KNOWLEDGE-BASED approach."

Expert systems grew out of this shift in emphasis.

An expert system is an AI program whose performance (1) depends more on the explicit presence of a large body of knowledge than on the incorporation of ingenuous computational procedures and (2) is intended to rival that of human experts. Human experts are characterized as possessing extensive knowledge about a narrow class of problems. Ill-defined and unstructured problems involving diagnosis or planning are problems typically thought to require human expertise to solve. Computer programs, however, can be supplied with a smaller, yet sufficient amount of knowledge to perform "expert" tasks efficiently because of the narrow bounds of many problems. Such programs, known as expert systems, can be useful in many areas such as education (to tutor, explain, test), business (to manage, plan, schedule, advise, forecast), and medicine (to diagnose, analyze, categorize)[89].

An expert system will assist a person with some knowledge in an area to develop an outline showing the steps needed to reach a solution for a given problem. Expert systems should ideally permit experts to test the system for completeness and correctness continually, enabling them to see how and why the knowledge was (or was not) applied. Ideal systems also allow experts, most of whom do not fully understand programming and computers, to make corrections and additions where necessary[90]. The ability to ask questions to lead a novice to a solution is another important part of any expert system, since the end user of the system , in all likelihood, will not be an expert.

Perhaps the easiest way to distinguish between expert systems and ordinary computer programs is to look at the way each organizes its knowledge. Most computer programs contain only two levels, data and program[91]. For example, the program (like a spreadsheet) takes detector readings (the data) and calculates a concentration for each reading, generating a report containing a calibration curve. In order to change the results, one would need to go into the spreadsheet and make changes to the data itself, either editing formulas or moving blocks of data within the spreadsheet.

Expert systems contain three levels, data, the knowledge base, and control structure[91]. The data still

consists of the detector readings, but the knowledge base contains rules about how to calculate concentrations, how to develop a calibration curve, and how to detect if the instrument is drifting. The control structure knows how to use the rules to interact with the data in order to determine answers to questions such as "Is the baseline of the instrument drifting?". These levels provide a convenient window to look at the ideas behind how an expert system works. For a more in depth look at expert systems, the reader is directed to several easily understood reviews in the literature[92,93,94].

The data level contains declarative knowledge (facts) about the problem being dealt with. They can either be general facts about the system (the path a solute takes flowing through a chromatographic system), or information that has been determined in running the system (current flow rates, a collection of readings from a detector, etc.). In the latter case, the data may be continually changing as additional information is received.

The knowledge base level contains the information specific to the particular type of problem being addressed. This information is often represented as IF-THEN rules, but other forms such as frames and semantic networks are also used (see the review articles mentioned above). In effect, rules contain the knowledge about how the facts are

interrelated, and how to perceive a solution. Rules are used to represent both heuristics and information that is generally agreed upon by "experts" within the field. Heuristics are general "rules of thumb" that expert use in solving problems, and often use confidence factors to indicate the amount of uncertainty that they contain.

Finally, the control structure is the computer program that decides how to use the knowledge in the knowledge base to arrive at a solution. Using the calibration example above, the user of the expert system would inquire if the instrument needs to be recalibrated. The control structure would look at the rule defining calibration, find that it needs to construct a calibration curve from the data, and then compares it to a previous calibration curve to see if it is within an allowable tolerance. If the data falls within the tolerance, the control structure would respond that no calibration is needed.

If one judges expert systems strictly in practical terms, the successes thus far have been very promising. Expert systems are just beginning to make a large impact in many areas such as Medical Science (MYCIN, Internist, etc.) and chemistry (Dendral, ECAT, Musch and Massart, etc.). In addition, many successful systems have not been reported in the current literature since companies consider them to be confidential. Many can be characterized as relatively small

systems (a few hundred rules at most) that solve specific problems, and in some cases result in substantial savings for the companies that invest in the technology. In order for these systems to be widely used by people who are not computer oriented, they must be able to respond in English (or a suitably close form) and be able to answer why and how it came up with its information. Most people are skeptical about "black boxes", especially when it comes to important areas like their health. Thus even though numerous hurdles have yet to be crossed, many questions have already been answered. These successful applications to practical problems will insure the continued interest in expert systems[88].

There are a number of works consisting of expert systems in the scientific arena[95,98,99]. The first of these contains two excellent reviews, one dealing with expert systems in general[96], and the other discussing development objectives and success factors for expert systems[97]. More concerted efforts towards developing artificial intelligence and expert systems applications in the area of chemistry are beginning to be reported. This was especially evident at two ACS national meetings where the Division of Computers in Chemistry sponsored symposia on these topics. A symposium on Applications of AI in Chemistry was held in Chicago in 1985[98], and a symposium on Expert System Applications in

Chemistry was held in Los Angeles in 1988[99]. Hohne and Pierce give an excellent overview of expert systems in the introduction to the resulting book. The International Symposium on Automation, Robotics, and Artificial Intelligence Applied to Analytical Chemistry, held in 1991 and 1993 in Montreux, France, also warrants noting. The proceedings from the 1993 meeting will be published in an upcoming issue of The Journal of Chemometrics and Intelligent Laboratory Systems.

To date, most of the successful expert systems in chemistry have been in the area of data analysis.[100] Dendral, an expert system that determined structure from mass spectra data, is a leader both in the area of expert systems applied to chemistry, and in the field of AI. Other areas of current work involve reaction synthesis and design, and instrument control and optimization. Several of these systems warrant mentioning. The reader is also directed to the symposia books mentioned above, as well as several pertinent journals[101].

DENDRAL is a chemistry expert system that is designed to take a spectrum from a mass spectrometer and predict the molecular structures that could account for the data. As a result of this effort, an algorithm was developed that inputs mass numbers and intensity pairs from a mass spectrogram, and applies heuristics to develop an initial

set of possible molecular structures for the data. These possibilities are further refined using additional constraints, until the resulting spectra produce spectra that closely match the initial data. Dendral was programmed directly in LISP, and took some 15 man years to complete[102].

The area of experimental design techniques has been targeted for expert system development by Deming, Palasota, Lee, and Sun[103]. They pose that the heavy computerization of instrumentation in the 1970's led to a crisis of sorts because of the lack of a systematic strategy for analytical method development. Out of this came three general guidelines for success: obtain a response, improve on it, and understand what was observed. These guidelines resulted in research into such areas as defining chromatographic performance, optimization strategies, and applying experimental design in such a was as to investigate the operational nature of chromatographic variables.

Deming et al. points out that current optimization techniques based on classical response surface mapping operate well during the early stages of searching for global optimum. However, gradient search techniques, such as sequential simplex, are probably more useful for "fine-tuning" of multiple factors in a method. This would most likely occur during the later stages of designing a

chromatographic method when attempting to achieve truly optimal performance.

II.B.7. Expert System Languages & Shells

The construction of an expert system, as well as any other program, requires tools. The construction tools available today for developing expert systems cover a broad range of complexity and price. These tools range from programming languages to expert system shells, which are a collection of utilities that a knowledge engineer has available in developing an expert system. Beneath all expert systems is a programming language. Although it has been proven that an expert system can even be written in BASIC[104] (a rule-based identification program), most real world problems suited to expert system solutions require special languages such as LISP, Prolog, or OPS5. These special languages provide the user with such things as list and other non-numeric processing statements, incremental compilation, and other prototyping facilities that enhance speedy development times by requiring the user to re-compile only the parts (or functions) of a file that were changed.

The language that has provided the backbone for AI research in the United States is LISP, which is an acronym for LISt Processing. This language is the product of a combination of list processing, lambda calculus, the ability

to define functions recursively (that is, in terms of itself), and an ability to define new functions, thereby expanding the language. John McCarthy joined these ideas in an entirely new approach to create this powerful tool for the field of AI. It has been claimed that all other languages used in AI research have their origins in LISP. At the core of LISP is lists, and everything is defined as lists of elements. This is how "Lots of Irritating Single Parenthesis" became another acronym for LISP[105]. A computer's memory can be viewed as a list of addressable memory cells, a program as a list of instructions, and data as a list of data objects consisting mostly of numbers and characters. Since each element is connected to the following element through the use of a pointer, LISP is often viewed as a pointer oriented language. Because both the data and program use the same representation, LISP programs can be written that will output other LISP programs, just as it would output data.

LISP is a flexible language, and in some ways resembles FORTH (or vice-versa) in that there are only a few basic functions, and all other functions are defined in terms of these basic functions. The user can create new functions to satisfy his or her own special needs. In fact, one can make as many levels as one wishes, even going so far as to construct an operating system using LISP. This ability to

be molded into other higher level languages is a great asset of LISP, as can be seen at Stanford in the Heuristics Programming Project, where they are working at creating higher-level knowledge representations for expert systems[105].

The ability to mold LISP has lead to many different dialects, some of which are Basic (Common) LISP, MacLISP, INTERLISPD, QLISP, ZETALISP, POP, and many more. In fact, some dialects have been changed so much that they are referred to as languages. Examples of this are Planner and Conniver[91].

Prolog now has appeared as the second language for AI research, mainly used in Europe, Japan and Australia. Its popularity has been growing since a very efficient Prolog compiler was installed at the University of Edinburgh's DEC 10 in 1977 by David Warren. Prolog stands for "PROgramming in LOGic" or "PROgramming language for LOGic", and was developed in 1972 at the University of Marseilles in France by Alain Colmerauer and his associates. Prolog is based on a simplified version of predicate logic. It has been gaining popularity in the United States ever since Japan choose Prolog as the language for its fifth-generation supercomputer project. In fact, a recent review[96] reported that Prolog was used in almost twice the number of applications as was the next most popular language, and that

it was chosen three to one over LISP as an application environment.

Prolog is designed for symbolic computation, yet is still efficient at list processing. It has the advantage of immediately attempting to return an answer in response to a query. It can be said that Prolog uses controlled logical deduction as its computational technique. Prolog, using logical inference for a control structure, tries to deduce if any specific conclusions can be reached from the stated facts. The use of a relational data base structure to store facts, or clauses, entered by the user is one of Prolog's greatest strengths. When all the facts have been entered, the user poses a query, or question, for Prolog to try to answer. In trying to deduce an answer, Prolog follows the same basic search pattern, moving through all of the clauses in a top-down and a left-to-right fashion. When a match is found, preconditions of the clause become the new query that Prolog must solve. When an answer for one condition is found, Prolog will repeat the process and continue to exhaust all possibilities for matches until solutions for all conditions are found. Because of this type of searching, the order in which your information appears in the data base determines the amount of time Prolog will take to find an answer. Hence, great thought must go into

building the data base[105]. Chapter VI goes into a more indepth discussion on the use of Prolog within CHESS.

The syntax of conventional programming languages of compatible power are much more complex than that of Prolog. The syntax of Prolog does include some operations that are not strictly logical (such as input and output), but these non-logical procedural aspects are kept to a minimum, allowing Prolog to be thought of as strictly a logical language[106].

The OPS language family grew out of the hypothesis made by Newell and Simon that most of human cognitive behavior can be sufficiently modeled using production rules. The expert system R1, used to configure VAX systems, was the first use of OPS5 in the field of knowledge engineering. OPS5 is not based on any specific control strategies or representation schemes, but uses general mechanisms which allow it to be molded to accommodate the characteristics of a specific domain. This permits different parts of the program to use different organization principles. Typically, systems developed using OPS5 represent rules in the familiar form "IF antecedent THEN consequence", and implement a forward chaining mechanism as their control structure. Many OPS5 programs place the burden of solving problems on the pattern matcher rather than on a search technique, utilizing the powerful pattern matching

capabilities of OPS5. The cost of maintaining this generality is the lack of a sophisticated programming environment. There is no built-in problem solving strategies or system organization, and no natural language interface that would allow a novice to communicate with OPS5.

There are other languages used for developing expert systems. Languages such as PSG II and RITA are rule based languages (similar to Prolog and OPS5), and still others have been or are being developed to handle semantic networks and frames. Examples of these languages are KRL, FRL, NETL, and Klone[91].

The rest of the spectrum of tools for building expert systems besides programming languages are normally considered expert system shells. These shells provide the expert system programmer with a variety of utilities such as predefined knowledge representations, control structures, debugging aids, and knowledge base construction and maintenance techniques. There are a wide variety of shells commercially available, each having there own strengths, weaknesses, and characteristics that make them unique. In each of the small passages that follow, a separate tool will be analyzed, giving information such as what types of problems it can solve, techniques it uses to solve them, and what machines the tool is available for. The tools presented

were chosen among those in wide use in academia and industry. It should be noted that no attempt has been made to give a complete review of the tools presented, and if further information is required, the reader is directed to the references listed with each tool.

In an attempt to reduce the high cost of a new system, developers began using the data level and control structure level of an existing system, building into them a knowledge base for the new problem. This significantly reduced the amount of time and effort required to build an expert system. The removal of the entire knowledge base from the expert system creates what is referred to as an expert system shell that can be used to construct other expert systems.

One such expert system shell is called GUESS, which stands for General pUrpose Expert System Shell.[107] It was developed at Virginia Tech in the Computer Science department by Mr. Lee and Professor Roach. This shell uses surface (attribute - object - value) and deep (knowledge frame) representations, both action frames and rules, and has the ability to utilize backward- and forward-chaining.

The present work describes an expert system shell for methods development in analytical chemistry. It is the authors belief that there is a body of general chemical knowledge that will be helpful in developing analytical

methods in virtually any area. Therefor this knowledge, involving linear free energy relations, additive relations in spectroscopy, and how to symbolically reason about mechanisms, should be left within the shell for use with other systems. What needs to be developed is the connections between this shell and new knowledge. Once the methodology to do this has been developed, new knowledge bases can be added that assist in solving problems in any area faced by the analytical chemist.

Even with an expert system shell, it is still difficult to develop an expert system. Abstracting the knowledge from the expert and incorporating it into the knowledge base is by far the most difficult part of the process. However, application of Personal Construct Theory shows great promise in greatly reducing the difficulty of this task as we shall see.

With this background, the reader can view the elements shown in Figure 1 that comprise the heart of the present research. CHESS is an expert system shell composed of functional group recognition, chemical structure searching algorithms, a molecular property evaluator, and automated knowledge acquisition capabilities. The application of this shell to chiral chromatography has lead to the development of the expert systems OPTIMER and CHIRULE.

III. CHESS's General Chemical Knowledge

The incorporation of general chemical knowledge into the core of the CHESS expert system shell is a main theme of this work. This knowledge is then accessed by the expert system application by making program calls to a set of standard chemical functions such as molecular property estimators and functional group recognition routines. These are the hooks that can be accessed by any expert system developed under CHESS.

The reader may have noticed the judicious use of the term "knowledge" and "knowledge base" throughout this work. Some may think of the information that is utilized by CHIRULE as simply data, and that a simple database lookup is sufficient to obtain an answer. The differences between data, information, and knowledge, as thought of in terms of a natural progression, will demonstrate why this is not the case. Using an analytical instrument as an example, the data is simply the raw numbers that are obtained from the instrument, whether they represent absorbance values or concentrations. In an effort to understand what the data means, a calibration curve is constructed, thus turning the data into information. Taking a series of these calibration curves, perhaps over a period of a day or a week, to determine if the instrument has begun to drift and is in

need of re-calibration, converts the information into knowledge.

These definitions of data, information, and knowledge can be used to illustrate the differences in the CHIRULE and OPTIMER systems (see Figure 6). Both systems were built using the CHESS expert system shell, which provides the hooks to access the reasoning processes based on chemical structures and properties. Three topics can be used to distinguish between the two systems: the approach used to store the knowledge, the type of information stored, and the technique used to gain access to that information.

OPTIMER approaches the situation by looking at data and even information and storing it in a traditional database style (a spreadsheet format). CHIRULE, on the other hand, uses some information but mostly knowledge in the form of rules and facts, storing them in a knowledgebase format. Database techniques such as simple lookups are sufficient for OPTIMER to gain access to the necessary data. CHIRULE takes advantage of sub-structure searching (SSS) and similarity searching in its quest to suggest a chiral column to serve as a starting point in developing a chiral chromatographic method.

This natural progression of taking the raw numbers, or data, turning it into calibration curves, or information, and finally comparing those curves in an effort to see if

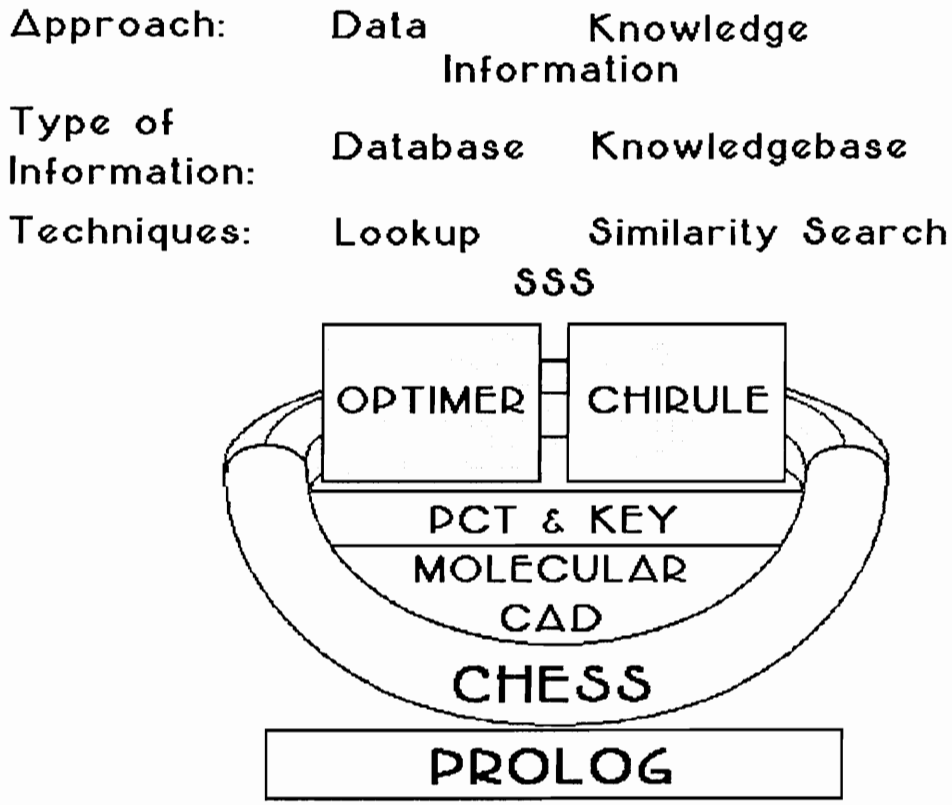


Figure 6: Differences between CHIRULE and OPTIMER

the instrument is drifting, illustrates the difference between simple data and knowledge. It also demonstrates how a scientist uses his or her knowledge to look at the data from the instrument and convert it first into information, and then into knowledge about how the instrument is operating.

In much the same way, CHESS provides the capabilities to input data involving molecular representation, convert it into information based on the functional groups that are contained within the molecule, and then transform it into knowledge by basing decisions on molecular properties. Further elaboration into each of the stages used in this progression follows.

III.A. Molecular Representation

The standard used for all functional group searches is based on the MOL file originally proposed by Wipke and Dyott[108], and used by Molecular Design Limited (MDL) in its MACCS and ISIS products. Text Frame 2 shows a MOL file representation of the molecule Phenylglycine. It can be divided into three sections: a header section, a list of atoms and a list of bonds. The header section contains, among other information, the name of the molecule, the package used to generate the MOL file, a comment line (blank in this example), and a line of numbers, the first two of

which represent the number of atoms and bonds respectively. The second section is made up of a list, in numerical order, of all of the non-Hydrogen atoms found within the molecule. The atom's x y z coordinates, the atomic symbol, and some stereochemical descripters are also included. The last section is a list of all bonds found within the molecule, with the atom numbers (as defined in the second section) of the two atoms terminating the bond being followed by the bond order and other stereochemical descripters. The reader is directed to the literature for a complete description of the MOL file format[109].

```
PHENYLGLYCINE
SSMACCS 5 04299200102D 1   0.00000     0.00000   0  SYS

 11 11  0  0  0
    3.5654    0.6179    0.0000 C   0  0  0  0  0
    2.5211    1.7494    0.0000 C   0  0  0  0  0
    1.0204    1.4452    0.0000 C   0  0  0  0  0
    0.5367   -0.0249    0.0000 C   0  0  0  0  0
    1.5810   -1.1563    0.0000 C   0  0  0  0  0
    3.0817   -0.8522    0.0000 C   0  0  0  0  0
   -1.8521    0.8618    0.0000 C   0  0  0  0  0
   -3.3755    0.7173    0.0000 O   0  0  0  0  0
   -1.2293    2.2567    0.0000 O   0  0  0  0  0
   -0.9641   -0.3920    0.0000 C   0  0  3  0  0
   -1.4708   -1.8988    0.0000 N   0  0  0  0  0
  1  2  1  0  0  0
  2  3  2  0  0  0
  3  4  1  0  0  0
  4  5  2  0  0  0
  5  6  1  0  0  0
  6  1  2  0  0  0
  7  8  2  0  0  0
  7  9  1  0  0  0
  7 10  1  0  0  0
 10  4  1  0  0  0
 10 11  1  0  0  0
```

Text Frame 2: MOL File of Phenylglycine

MOL files are directly read into CHESS and converted into CHESS's Molecule Knowledge Base (or KB). This knowledge base is used for all reasoning processes that are based on chemical structure, such as atom and functional group searches. However, the MOL file is not the only representation that CHESS utilizes.

As indicated previously, SMILES strings are used as the basis for all molecular searches. This is done because the SMILES strings used are unique, whereas chemical names are not. Text Frame 3 shows several examples of SMILES strings. There are several things to note about the format of SMILES strings: Aromatic Carbon and Nitrogen atoms are lower case, "=" represents a double bond, numbers are used to close ring structures (the two "c1"s represent ring closure in the phenyl groups), and parentheses are used to group bonded atoms. For a complete description of the rules for generating SMILES strings, the reader is directed to the literature[54].

PHENYLGLYCINE	"c(ccc1C(N)C(=O)O)cc1"
D-PHENYLALANINE	"c(ccc1CC(N)C(=O)O)cc1"
MANDELIC ACID	"c(ccc1C(O)C(=O)O)cc1"

Text Frame 3: Sample SMILES Strings

```
 20 ATOMS   20 BONDS     0 CHARGES L
  1 CAR      0.7646     1.2452    0.4533    0.0000
  2 CAR      2.1556     1.2452    0.4533    0.0000
  3 CAR      2.8559     2.4796    0.6662    0.0000
  4 CAR      2.1569     3.6140    1.2791    0.0000
  5 CAR      0.7650     3.6141    1.2791    0.0000
  6 CAR      0.0678     2.4797    0.6662    0.0000
  7 -        2.7005     1.0626    0.1321    0.0000
  8 -        3.9399     2.8794    0.8661    0.0000
  9 -        2.7004     4.8960    1.6001    0.0000
 10 -        0.2229     4.8960    1.6001    0.0000
 11 -       -1.0161     2.8794    0.8661    0.0000
 12 C3       0.0000     1.0000    0.0000    0.0000
 13 -        0.6445    -1.6454   -0.4053    0.0000
 14 C2      -0.6840    -1.6336    1.2131    0.0000
 15 O2      -1.9088    -1.5942    1.3189    0.0000
 16 O3       0.0481    -1.8257    2.1560    0.0000
 17 -       -0.5366    -1.6666    2.8378    0.0000
 18 N3      -1.0492    -1.4099   -1.0357    0.0000
 19 -       -1.3914    -1.4109   -1.4941    0.0000
 20 -       -0.6260    -1.1217   -1.7034    0.0000
  1    1     2  AROMATIC
  2    2     3  AROMATIC
  3    3     4  AROMATIC
  4    4     5  AROMATIC
  5    5     6  AROMATIC
  6    6     1  AROMATIC
  7    1    12  SINGLE
  8    2     7  SINGLE
  9    3     8  SINGLE
 10    4     9  SINGLE
 11    5    10  SINGLE
 12    6    11  SINGLE
 13   12    13  SINGLE
 14   12    18  SINGLE
 15   12    14  SINGLE
 16   14    15  DOUBLE
 17   14    16  SINGLE
 18   16    17  SINGLE
 19   18    19  SINGLE
 20   18    20  SINGLE
```

Text Frame 4: SPARC MOL File Format

Rather than developing a new SMILES string generator, the one utilized by the SPARC program, entitled CONVB, was employed. CONVB takes the MOL file of a molecule as input,

converts it, and writes a file containing its corresponding SMILES string. CONVB is usually accessed by MEDIT, the molecular editor that comes with SPARC. One of the menu choices within MEDIT is "SMILES", which will display the SMILES string for the molecule currently being drawn. However, the format of the MOL file used by MEDIT is slightly different than the one utilized by MDL's software, and is shown for Phenylglycine in Text Frame 4. Compare this with MDL's MOL file shown in Text Frame 2.

When choosing "SMILES" from the MEDIT menu, CONVB is run by MEDIT to generate the SMILES string of the molecule in the current molecular buffer. This is accomplished by writing a Hydrogen deficient MOL file to the file name TEST.SS in the directory C:\SPARC\MEDIT\MOL\, then spawning a process that runs the program C:\SPARC\MEDIT\CONVB, which generates the file "C:\SPARC\SS\TEST.SS" containing the SMILES string of the molecule. The Hydrogen deficient MOL file is shown in Text Frame 5. Compare this

```
  11 ATOMS    11 BONDS    0 CHARGES L

   1 CAR     0.7648   1.2452   0.4533     0.0000
   2 CAR     2.1588   1.2452   0.4533     0.0000
   3 CAR     2.8559   2.3796   0.8662     0.0000
   4 CAR     2.1589   3.5140   1.2791     0.0000
   5 CAR     0.7650   3.5141   1.2791     0.0000
   6 CAR     0.0678   2.3797   0.8662     0.0000
   7 C3      0.0000   0.0000   0.0000     0.0000
   8 C2     -0.6840  -0.6336   1.2131     0.0000
   9 O2     -1.9088  -0.5942   1.3189     0.0000
  10 O3      0.0481  -1.2287   2.1560     0.0000
  11 N3     -1.0492   0.4093  -1.0357     0.0000
   1     1     2  AROMATIC
   2     2     3  AROMATIC
   3     3     4  AROMATIC
   4     4     5  AROMATIC
   5     5     6  AROMATIC
   6     6     1  AROMATIC
   7     1     7  SINGLE
   8     7    11  SINGLE
   9     7     8  SINGLE
  10     8     9  DOUBLE
  11     8    10  SINGLE
```

Text Frame 5: Hydrogen Deficient MOL File Input for CONVB

with the MEDIT MOL file shown for Phenylglycine in Text Frame 4. CHESS handles the necessary conversion between MDL's MOL file and the Hydrogen deficient MOL file used by CONVB (see predicate "smiles_str" in module SMILES.HC in Appendix F).

III.B. Functional Group Recognition

The capability to reason based on molecular structure is obtained by querying the internal MOL file representation for various functional groups. The querying process takes advantage of the "hooks" built into CHESS. CHESS has two different sets of hooks that are available to the developer: one set that determines only the presence or absence of a functional group, and another set that enables the developer to determine exactly where the functional group is within the molecule.

The first set of hooks are those that can be accessed through KEY. All of them involve a single parameter, that of the name of the molecule. This name is determined when KEY begins to run a rulebase and asks the user for a target molecule (see OPTIMER example, Text Frame 25, Chapter IV.B), and is handled by the KEY inference engine by predicates "ask_user" and "exrule" (see Appendix E, Inference Engine for VPI Prolog). Table III lists definitions for all of these hooks currently built into the CHESS shell. In order

Table III: CHESS Functional Group "Hook" Queries

"CdC ?name"	- Searches for alkenes
"CtC ?name"	- Searches for alkynes
"CtN ?name"	- Searches for nitriles
"Oeth ?name"	- Searches for ether linkages
"OsH ?name"	- Searches for alcohols
"CdO ?name"	- Searches for carbonyls
"CHdO ?name"	- Searches for aldehydes
"X2CdO ?name"	- Searches for ketones
"COOH ?name"	- Searches for carboxylic acids
"COOX ?name"	- Searches for esters
"N3 ?name"	- Searches for amines
"NT ?name"	- Searches for tertiary amines
"NS ?name"	- Searches for secondary amines
"NP ?name"	- Searches for primary amines

to take advantage of them, the developer would use the specified name of the functional group desired (omitting the "?name"), and utilize it as the name [or left-hand pole of that trait, (cf. the list of traits for OPTIMER in Chapter IV, Text Frame 16)]. When the KEY rulebase is executed, the KEY inference engine will utilize CHESS's hooks to determine if that functional group is present.

Denoting a central atom within the functional group, as described by Esack and Bersohn[60, 61], is the basis for the second set of functional group recognition routines made available by CHESS (see section II.B.2). It defines which atom within the functional group will be utilized for performing all searches. Table IV lists the central atom definitions utilized by CHESS.

All procedures to determine the presence of a functional group are written as Prolog predicates. Prolog refers to subroutines as predicates (see Chapter VI for a further discussion on Prolog). In fact, if one were to look at all of the predicates defining the first set of functional group recognition routines, each of then makes use of its counterpart in the second set, but does not make use of the last two parameters (see Module FG.HC in Appendix F).

There are three parameters for each of these predicates: ?name is the name of the molecule which is to be

Table IV: Functional Group Central Atom* Definitions

$-C\equiv N^*$	- Nitriles
$-C-O^*-C-$	- Ether linkages
$C-O^*-H$	- Alcohols
$X_2C^*=O$	- Carbonyls
$-C^*H=O$	- Aldehydes
$R_2C^*=O$	- Ketones
$-C^*=OOH$	- Carboxylic acids
$-C^*=OOR$	- Esters
X_3N^*	- Amines
R_3N^*	- Tertiary amines
R_2N^*H	- Secondary amines
RN^*H_2	- Primary amines
$-C=C-$	- Alkenes[1]
$-C\equiv C-$	- Alkynes[1]

R denotes any non-hydrogen atom
X denotes any atom
[1]Symmetric functional group, no central atom defined

Table V: Functional Group Set II Definitions

```
"has_CdC ?name ?A1 ?A2"   - Searches for alkenes
"has_CtC ?name ?A1 ?A2"   - Searches for alkynes
"has_CtN ?name ?A1 ?A2"   - Searches for nitriles
"has_Oeth ?name ?A1 ?A2"  - Searches for ether linkages
"has_OsH ?name ?A1 ?A2"   - Searches for alcohols
"has_CdO ?name ?A1 ?A2"   - Searches for carbonyls
"has_CHdO ?name ?A1 ?A2"  - Searches for aldehydes
"has_X2CdO ?name ?A1 ?A2"- Searches for ketones
"has_COOH ?name ?A1 ?A2"  - Searches for carboxylic acids
"has_COOX ?name ?A1 ?A2"  - Searches for esters
"has_N3 ?name ?A1 ?A2"    - Searches for amines
"has_NT ?name ?A1 ?A2"    - Searches for tertiary amines
"has_NS ?name ?A1 ?A2"    - Searches for secondary amines
"has_NP ?name ?A1 ?A2"    - Searches for primary amines
```

tested (the only parameter used in common with the first set of predicates), ?A1 is the atom number of the central atom as defined in Table IV, and ?A2 is an atom number whose meaning depends on the particular functional group predicate being utilized. When the predicate is invoked, typically only the molecule name is provided in ?name. The predicate will then return the atom numbers of the appropriate atoms in ?A1 and ?A2. Table V shows these functional group definitions.

These definitions allow the developer to make queries that determine relative locations of functional groups. For example, suppose a developer wants to determine if there were two hydroxy groups in an alpha position to each other. From the definition of "has_OsH" in Module FG.HC in Appendix F, we find that ?A2 is the atom number of where the hydroxy group is attached to the molecule. Using this information, the predicate "alpha_OH" shown in Text Frame 6 can be written. After determining the bond list for the target molecule, two hydroxy groups are found (making sure they are not the same group). Having them in the alpha position requires a bond between the atoms to which each hydroxy group is attached. If the predicate succeeds, it

```
((alpha_OH ?name ?A1 ?A3) if
        (molecule ?name ? ? ?bdlist)
        (has_OsH ?name ?A1 ?A2)
        (has_OsH ?name ?A3 ?A4)
        (not (== ?A1 ?A3))
        (findsb ?bdlist ?A2 ?A4)
)
```

Text Frame 6: Using Functional Group Predicates

will return the name of the molecule in ?name, and the oxygen atom numbers of the two hydroxy groups in ?A1 and ?A3.

All functional group predicates are constructed out of lower level Prolog predicates developed to query the Molecule KB with respect to individual atoms and bonds. Text Frame 7 contains the Prolog code necessary to determine if a particular molecule contains a carbonyl functional group. The first line defines the Prolog predicate called "has_CdO" along with its 3 parameters. The first parameter, ?name, is the name of the molecule in which you want to search for a carbonyl group. "?A1" is the atom number of the central atom (the carbonyl carbon atom) as defined in Table IV, and "?A2" is the carbonyl oxygen atom. Typically "?A1" and "?A2" will not be specified initially, but will be returned after the predicate is executed.

```
((has_CdO ?name ?A1 ?A2) if
        (molecule ?name ? ?atlist ?bdlist)
        (is_atom ?atlist "O" ?A2)
        (finddb ?bdlist ?A1 ?A2)
        (is_atom ?atlist "C" ?A1)
)
```

Text Frame 7: Predicate for Determining Carbonyls

The next four lines represent the 4 tasks that will be accomplished by the predicate "has_CdO" (a C=O) and are in fact predicates themselves. The first is a query to the Molecule KB, and returns the atom list "?atlist" and bond list "?bdlist" for the molecule "?name". The atom and bond lists are roughly equivalent to what is found in the MOL

file of the molecule. The second predicate is looking to see if there is an oxygen atom within the atom list, its atom number being returned in ?A2. Next the bond list is examined to find a double bond (predicate "finddb") between the oxygen atom ?A2 and another atom ?A1. Finally, the identity of atom number ?A1 is confirmed to be a carbon atom.

The predicate for any particular functional group is thus reduced to a combination of determining that the required atoms and bonds are present. While there are other ways by which to construct the predicate "alpha_OH" defined in Text Frame 6, this method was selected because of its more efficient use of Prolog's search process. Refer to Chapter VI for a further explanation of additional capabilities gained by using Prolog and defining functional group predicates in this fashion. Table VI lists a number of the predicates used for queries concerning atoms and bonds within a molecule. This easily expandable syntax is imperative if other researchers are to be able to build on preceding work. Using these pre-defined routines, new functional group searching procedures can be constructed. Refer to Appendix F for all of the Prolog predicates used by CHESS: the module FG.HC contains the functional group recognition predicates, and module CHEM.HC contains all of the lower level predicates that look at atoms and bonds.

Table VI: Atom and Bond Determination Predicates

(findsb ?bdlist ?A1 ?A2) - Finds a single bond in the bond list ?bdlist between atom numbers ?A1 and ?A2.

(ofindsb ?bdlist ?A1 ?A2 ?A3) - Finds a second single bond in the bond list ?bdlist between atom numbers ?A1 and ?A3 other than the one that already exists between ?A1 and ?A2.

(finddb ?bdlist ?A1 ?A2) - Finds a double bond in the bond list ?bdlist between atom numbers ?A1 and ?A2.

(findtb ?bdlist ?A1 ?A2) - Finds a tripple bond in the bond list ?bdlist between atom numbers ?A1 and ?A2.

(is_bond ?bdlist ?A1 ?A2) - Determines if there is a bond in the bond list ?bdlist between atom numbers ?A1 and ?A2. Does not determine the type of the bond.

(bond_type ?bdlist ?A1 ?A2 ?type) - Determines the bond type between atom numbers ?A1 and ?A2 (1=single, 2=double, 3=triple).

(is_atom ?atlist ?atsymb ?Anum) - Determines if there is an atom number ?Anum in the atom list ?atlist with the atomic symbol ?atsymb. All parameters are typically supplied, as it is used to insure that atom number ?Anum is a particular element.

(attached_H ?atlist ?bdlist ?A1 ?A2) - Determines if there are any single bonds with atom ?A1 that are either not hydrogens or are found in the atom number list ?A2. It will fail if there are any bonds with non-hydrogen atoms that are not found in ?A2.

III.C. SPARC and Property Estimation

The capability to make decisions based on the values of various molecular properties is an important part of both CHESS and CHIRULE. Although databases of molecular properties are useful for any expert system, it is virtually impossible to make them complete. Thus the ability to calculate molecular properties based solely on molecular structure is imperative. To address this de novo goal, an expert system for the estimation of chemical and physical reactivity was incorporated into CHESS. SPARC[68] (SPARC Performs Automated Reasoning in Chemistry) has the capability to calculate properties based solely on the chemical structure of the molecule. The properties used in this work are average molecular polarizability, molecular volume, and Hydrogen-bonding descripters.

SPARC itself is a collection of programs providing an integrated environment to gain access to the property evaluator, a graphical molecular structure drawing tool (MEDIT), a SMILES string generator (CONVB), and a batch mode interpreter for performing property estimations based on information read in from a file. MACCS (rather than MEDIT) has been chosen for the input of molecular structures into the CHESS system because of its enhanced editing capabilities. The use of CONVB has already been described above.

The batch mode reader is used as follows. In using the interactive portion of SPARC, the user first inputs the molecule, then is presented with a menu of properties that can be calculated (see Chapter II.B.3 for a complete list of properties). Each property is calculated one at a time, displaying its results individually. In contrast, the batch mode reader allows the user to develop a file with multiple molecules and multiple properties to be calculated in the same pass automatically. The input file (C:\SPARC\BATCH.DAT), shown in Text Frame 8, consists of the properties to calculate (Hydrogen bonding donating and withdrawing capabilities, molecular polarizability, and molecular volume), the SMILES string of the molecule (Phenylglycine), and the conditions to use (default is 25°C).

```
['/25'].
'c(cccIC(N)C(=0)0)ccl'.
alpha.
['/25'].
'c(cccIC(N)C(=0)0)ccl'.
beta.
['/25'].
'c(cccIC(N)C(=0)0)ccl'.
pol.
['/25'].
'c(cccIC(N)C(=0)0)ccl'.
vol.
```

Text Frame 8: SPARC Batch Mode Input File

```
*** BATCH MODE OUTPUT ***

*** Processing Molecule Number 1
*** Smiles String: c(cccIC(N)C(=0)0)ccl
*** Property Type: Alpha
*** Solvent Mixture: N/A
*** Temperature: 25
*** Observed Value's: [/25]
*** Calculated Value: 0.754963562

*** Processing Molecule Number 2
*** Smiles String: c(cccIC(N)C(=0)0)ccl
*** Property Type: Beta
*** Solvent Mixture: N/A
*** Temperature: 25
*** Observed Value's: [/25]
*** Calculated Value: 1.069617675

*** Processing Molecule Number 3
*** Smiles String: c(cccIC(N)C(=0)0)ccl
*** Property Type: Polarizability
*** Solvent Mixture: N/A
*** Temperature: 25
*** Observed Value's: [/25]
*** Calculated Value: 1.620039642

*** Processing Molecule Number 4
*** Smiles String: c(cccIC(N)C(=0)0)ccl
*** Property Type: Molecular Volume
*** Solvent Mixture: N/A
*** Temperature: 25
*** Observed Value's: [/25]
*** Calculated Value: 123.5469827
```

Text Frame 9: SPARC Batch Mode Output File

SPARC, running in batch mode, then accepts this file and generates another file (C:\SPARC\BATCH.OUT) that contains the molecules and the calculated values for the desired properties. This file also contains the molecules SMILES string, the property type, the solvent mixture (if applicable), temperature, and any observed values that were provided for comparison in the input file (see Text Frame 9). This file is read in by CHESS, and the property calculations are stored in the Fragment Value KB.

The user can also request that SPARC create a "dribble" file (C:\SPARC\BATCH.DRB) where intermediate results are noted. This file contains the SMILES string of the molecule, any reactophores that are found within the molecule, the property, the solvent mixture (if applicable), the temperature, and any observed values that were noted in the batch mode input file. This file is shown in Text Frame 10, and is useful if errors occur during the calculations.

These are the steps that CHESS needs to perform in order to gain access to the molecular property calculation capabilities of SPARC. CHIRULE divides each molecule (in the Molecule KB) into four molecular fragments (represented int he Molecular Fragment KB) around the molecule's chiral carbon, and uses SPARC to calculate the molecular properties of each fragment (contained in the Fragment Value KB).

CHIRULE does a similarity search (using these values) and suggests a chiral column to use in developing a chiral separation method. CHESS generates the file that contains a list of the molecular fragment's SMILES strings along with the property to calculate. CHESS then processes the results file, adding the fragments' property values to the internal Fragment Value KB.

```
                *** BATCH MODE OUTPUT ***

*** Processing Molecule Number 1
*** Smiles String: c(ccc1C(N)C(=O)O)cc1
*** Building the atom-specs database.
*** searching for reactophores
*** the reactophores found are:
    arom1 of Type benzene
*** Upgrading All Gamas
*** Calculating Steric Twist

*** Property Type: Alpha
*** Solvent Mixture: N/A
*** Temperature: 25
*** Observed Value's: [/25]

*** Processing Molecule Number 2
*** Smiles String: c(ccc1C(N)C(=O)O)cc1
*** Property Type: Beta
*** Solvent Mixture: N/A
*** Temperature: 25
*** Observed Value's: [/25]

*** Processing Molecule Number 3
*** Smiles String: c(ccc1C(N)C(=O)O)cc1
*** Property Type: Polarizability
*** Solvent Mixture: N/A
*** Temperature: 25
*** Observed Value's: [/25]

*** Processing Molecule Number 4
*** Smiles String: c(ccc1C(N)C(=O)O)cc1
*** Property Type: Molecular Volume
*** Solvent Mixture: N/A
*** Temperature: 25
*** Observed Value's: [/25]
```

Text Frame 10: SPARC Dribble File

IV. Automated Knowledge Acquisition - KEY

In order to bring the technology of expert systems into the hands of scientists, a tool must exist that will enable them to interact with a computer through a dialog directed session, and develop a working system on their own. This will help remove many of the barriers that are often encountered with exhaustive interviewing, the typical method used for constructing expert systems. In addition, as the scientist interacts with the program, a sense of ownership develops as the knowledge is entered and refined. Researchers indicate that this feeling of ownership also helps to relieve the sometimes unarticulated fear of being replaced by "a computer".

KEY (Knowledge Extraction utilitY), based on Personal Construct Theory (PCT), was developed in an effort to address these issues. The most important task in showing how KEY is used to develop an application is choosing an appropriate example. Since not all types of knowledge are not adequately represented by the knowledge grid structure, this topic will be addressed first. Following this, the development of OPTIMER, an expert system built using KEY, will be discussed without covering the theory behind the methods employed. OPTIMER, constructed by a chromatographer, is an expert system that represents the

decision tree knowledge found in most chiral column selection guides. Depending on the structure of the isomers for which the user wants to develop a chiral separation, OPTIMER suggests where to search for molecule/column pairs similar to the users interest. This section serves as a guide for anyone interested in developing a chemical system using KEY.

Finally, the technical aspects of PCT and their influence on KEY will be discussed. To facilitate this discussion, it will be divided into the three individual phases of eliciting knowledge from the user: (1) the acquisition of the grid, (2) the grid analysis, and (3) the generation of the knowledge base. This is followed by a discussion of the data files and program structures used by KEY. These sections cover in detail how the knowledge grid is converted into a working expert system.

Note that the person who uses KEY to construct a system is termed a "developer" and the person running the finished system is referred to as the "user". All examples shown in the following sections are taken from the construction of the OPTIMER system. Appendix A contains a complete transcript of the session developing OPTIMER.

IV.A. Choosing an Appropriate Topic

What types of problems are appropriate to PCT? Studies have shown that PCT is most appropriately applied to problem areas that are analytic in nature, such as classification and diagnosis. Analytic problems can often be represented in the form of a decision tree, and formulating the problem into such a tree structure can considerably speed up the development process (see below). PCT does not effectively handle synthesis type problems where planning or design is required, or problems that require combining analytical and synthesis skills, such as management or monitoring.

Once a topic is chosen, it is best if some background work is completed prior to using KEY. This would be done by writing all the possible solutions on a piece of paper, along with a number of questions that would be asked in the process of deciding which of the solutions is most appropriate for any particular situation. This should be sufficient if the developer wishes to have KEY prompt for the entry of the traits by listing out 3 elements and asking for entry of an idea that distinguishes two of them from the third (vide infra).

However, as developers becomes more familiar with the system, they may wish to simply enter all of the traits. Remembering that the methodology is very applicable to decision tree type problems, the developer should draw out a

rough decision tree of the area for which they wish to construct an expert system. The decision points in the tree will become the traits, and the leaf nodes of the tree will become the elements.

IV.B. Example - OPTIMER

KEY is targeted towards assisting those not familiar with programming in the development of expert systems. To demonstrate this premise, a separations scientist[110] took the Daicel chiral column application guide[111] and used KEY to develop a system with applicability to his needs. The author provided only resource advice. The result was a system, similar to OPTIMER, that was able to take advantage of the chemical hooks made available in CHESS (refer to Appendix A for a transcript of the interaction). It was possible for the scientist to develop and test the system without becoming a computer programmer.

The decision tree already shown in the Daicel chiral column application guide (see Figure 7[111]) proved to be an appropriate topic for KEY. The questions contained within the diamond shaped nodes are the traits that will be used, and the elements are the page numbers of where to look for molecules similar to the user's target molecule. When this application is developed as an expert system, a user would input the structure of the isomers they are interested in

Application guide for column selection according to functional group.

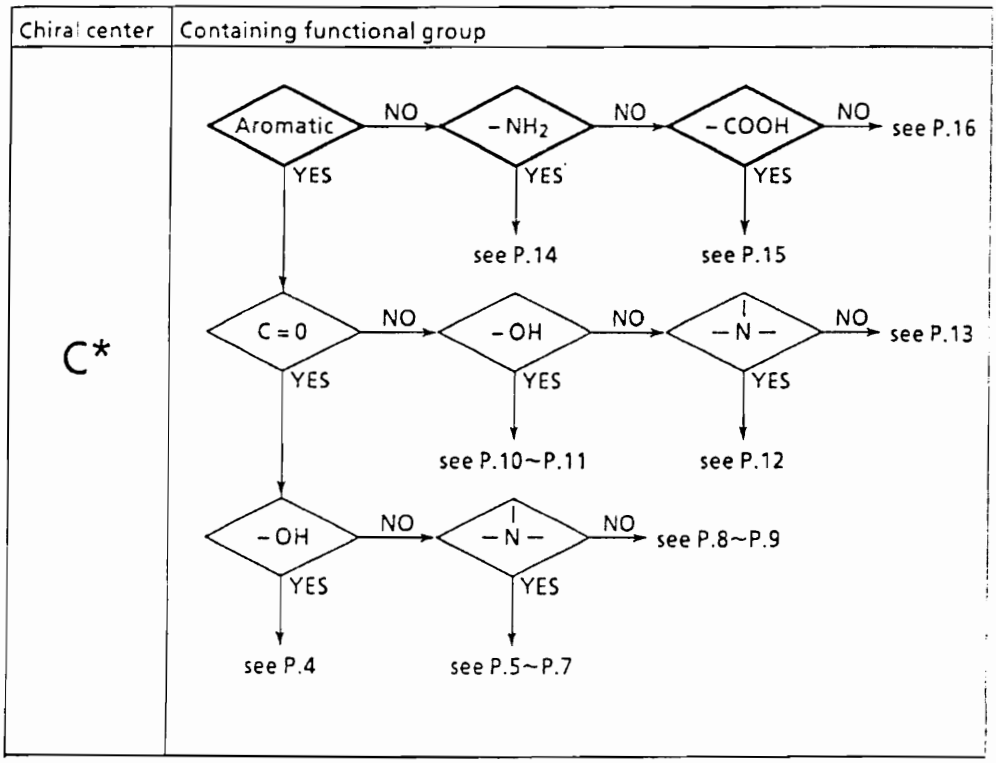


Figure 7: Daicel Chiral Column Selection Guide

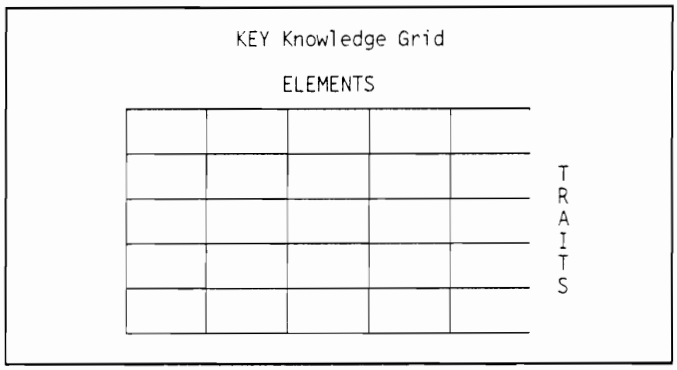
separating, and KEY will suggest where within the application guide to begin looking for a related molecule/column pair.

The application guide groups molecules with known separations techniques together according to functional groups. The decision tree shown in Figure 7 directs the user where to look for molecules similar to their own according to the functional groups it contains. By the judicious choice of trait names, the developer employing KEY takes advantage of the "hooks" built into CHESS. In the finished system, OPTIMER obtains answers to questions regarding functional groups by asking CHESS to look at the MOL file representation of the target molecule. The user of OPTIMER need not answer questions regarding functional groups found within their target molecule, only supply its MOL file.

This section does not explain in detail the calculations that are performed, nor does it cover all of the capabilities of KEY since they will be explained in later sections of this chapter. What follows is intended to be an example of how one would go through and use the tool to develop an application.

After starting KEY, acquiring the grid is the first task to be accomplished. The knowledge grid can be conceptually thought of as a spreadsheet like structure (see

Text Frame 11). The elements, representing the answers the system is to suggest, are placed along the top of the spreadsheet. The characteristics used to distinguish between them, known as traits, are placed along the right hand side of the spreadsheet. Filling in each elements value for all the traits in the spreadsheet completes the acquisition of the knowledge grid.



Text Frame 11: Spreadsheet format of Grid

OPTIMER's goal is to suggest where to go to look for a molecule that is similar to the user's molecule. Hence, the elements are the page numbers found within the application guide. In this case, it is a new system and the elements will need to be entered, so the developer types "ENTER" as directed by the KEY command menu (see Text Frame 12). Note that KEY uses a menuing system to interact with the developer. The developer selects which response (shown in capital letters)

```
Please enter the names of the elements.
        Your choices for obtaining the element names
        are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved element list
    ENTER - enter a new element list
    EDIT - edit the list of elements
    DEL - delete the list of elements
    SAVE - save the list of elements
    LIST - list out the elements
    DONE - finished entering element list
    HELP - print out this list of commands

Please enter your choice here -> ENTER
```

Text Frame 12: Element Entry Menu

is appropriate from among the selections and types it in at the prompt.

While entering the elements, the developer can list out or edit any of the elements entered thus far, continue entering elements, or finish entering the elements as shown by the menu selection (see Text Frame 13). For each of the elements, KEY will prompt the developer for a name and a three letter abbreviation (see Text Frame 13). They are entered one after another as prompted, typing DONE when all have been entered.

```
Now enter the names of the elements, one to a line, or the
        name of a command.

There are 3 commands you can enter, and they are:
    LIST - list out the elements
    DONE - finished entering element list
    HELP - print out this list of commands

Name of element 1 -> p 4
   Three letter abbreviation for element : p-4 -> 4

Name of element 2 -> p 5-7
   Three letter abbreviation for element : p-5-7 -> 5-7

Name of element 3 -> p 8-9
   Three letter abbreviation for element : p-8-9 -> 8-9

Name of element 4 -> p 10-11
   Three letter abbreviation for element : p-10-11 -> 10-11
```

Text Frame 13: Entering OPTIMER Elements

This will take the developer back to the menu for obtaining the list of elements. From here the developer can list out all of the elements by typing "LIST" to make sure no editing changes are necessary before proceeding (see

```
There are 9 elements currently.  Their names follow :

The name of element 1 is {p-4}. and its abbreviation is {4}.
The name of element 2 is {p-5-7}. and its abbreviation is {5-7}.
The name of element 3 is {p-8-9}. and its abbreviation is {8-9}.
The name of element 4 is {p-10-11}. and its abbreviation is {10-}.
The name of element 5 is {p-12}. and its abbreviation is {12}.
The name of element 6 is {p-13}. and its abbreviation is {13}.
The name of element 7 is {p-14}. and its abbreviation is {14}.
The name of element 8 is {p-15}. and its abbreviation is {15}.
The name of element 9 is {p-16}. and its abbreviation is {16}.

Please hit RETURN to continue -
```

Text Frame 14: OPTIMER Element List

Text Frame 14). After the developer is satisfied with all of the elements, typing "DONE" will move on to the menu for entering the traits.

The easiest way to enter a new trait list is to choose "ENTER" from the trait entry menu. After listing the valid commands the developer can type, KEY will begin to present the elements in groups of three (see Text Frame 15). The developer must suggest a trait that will distinguish two of the elements from the third and enter the trait, its opposite, and a description of the trait. If the developer cannot come up with a trait for the three elements presented, typing "NEXT" will move on to the next set of three elements.

```
Now enter the names of the traits, one to a line, or the
        name of a command.

There are 4 commands you can enter, and they are:
    LIST - list out the traits
    NEXT - go on to the next triad of traits
    DONE - finished entering trait list
    HELP - print out this list of commands

Consider the following three elements :
                    p-4
                p-10-11
                   p-12

Please determine a trait that two of the above elements
        have that the third does not, and enter the
        name of that trait below.

Name of trait 1 -> CdO
     Now please enter the opposite of CdO here -> no CdO
     Think of a name that can be used to describe this
        trait, and enter it here -> carbonyl
```

Text Frame 15: Entering Traits Using Triads of Elements

In order to take advantage of the hooks built into the CHESS shell, the developer must choose the specific trait names that are already understood by CHESS. The hooks currently revolve around the determination of functional groups within a molecule. The advantage of using the hooks is that CHESS will determine the presence of functional

groups rather than querying the user of the finished system. Table III shows a list of the trait names that utilize the available hooks. The developer is not, however, limited to using these trait names. For any traits that KEY does not find hooks within CHESS, KEY will ask the user for the traits value at execution time. Refer to Section III B for a more complete description of the functional group hooks that are available.

This process continues until the developer is satisfied that the traits entered are sufficient, at which time typing "DONE" will return them to the menu for obtaining the traits. Keep in mind that the analysis phase will allow for the addition of more traits to eliminate ambiguities within the knowledge grid. Listing out the traits by typing "LIST" (see Text Frame 16) can assure that there are no editing changes that must be made before the grid is constructed. Typing "DONE" will move on to the final stage of grid acquisition, that of filling in the grid strengths.

```
Please enter your choice here -> list

There are 6 traits currently.  Their names follow :

The name of trait 1 is {CdO}, its opposite is {no-CdO},
    and a description of this trait is {carbonyl}.
The name of trait 2 is {has-benz}, its opposite is {no-benz},
    and a description of this trait is {aromatic-ring}.
The name of trait 3 is {OsH}, its opposite is {no-OsH},
    and a description of this trait is {hydroxyl}.
The name of trait 4 is {N3}, its opposite is {no-N3},
    and a description of this trait is {amine}.
The name of trait 5 is {NP}, its opposite is {no-NP},
    and a description of this trait is {primary-amine}.
The name of trait 6 is {COOH}, its opposite is {no-COOH},
    and a description of this trait is {carboxylic-acid}.

Please hit RETURN to continue -
```

Text Frame 16: OPTIMER Trait List

The development of a new grid involves filling in the spreadsheet (see Text Frame 11) with numerical values that represents the strength of each element with respect to each trait. After typing "ENTER" to enter the grid values, the developer is presented with each element, one at a time, and asked to rate that element individually against each trait (see Text Frame 17). Remember that the traits are bi-polar constructs, and have left and right poles.

```
Please rate each of the following elements on a scale of 1 to 5,
     where {CdO} is 5 and {no-CdO} is 1.  (type any negative
     number to exit, or '0' to remain unchanged)

Value for p-4 is currently 0, new value is -> 5
Value for p-5-7 is currently 0, new value is -> 5
Value for p-8-9 is currently 0, new value is -> 5
Value for p-10-11 is currently 0, new value is -> 1
Value for p-12 is currently 0, new value is -> 1
Value for p-13 is currently 0, new value is -> 1
Value for p-14 is currently 0, new value is -> 3
Value for p-15 is currently 0, new value is -> 3
Value for p-16 is currently 0, new value is -> 3
```

Text Frame 17: Entering Grid Values

The poles can be thought of as extensions of "true" and "false". The value entered represents where the element falls between the two poles of the trait on a scale of 1 to 5. In this case, 5 means that the element possesses the right pole, 1 means the left pole, and 3 means the element does not necessarily posses either pole

```
Please enter your choice here -> list

   4 = p-4
     5-7 = p-5-7
         8-9 = p-8-9
             10- = p-10-11
                 12 = p-12
                     13 = p-13
                         14 = p-14
                             15 = p-15
                                 16 = p-16
  4 5-7 8-9 10-  12  13  14  15  16     trait pairs
5   5   5   1   1   1   3   3   3 CdO = 5, no-CdO = 1
5   5   5   5   5   5   1   1   1 has-benz = 5, no-benz = 1
5   1   1   5   1   1   3   3   3 OsH = 5, no-OsH = 1
3   5   1   3   5   1   3   3   3 N3 = 5, no-N3 = 1
3   3   3   3   3   3   5   1   1 NP = 5, no-NP = 1
3   3   3   3   3   3   3   5   1 COOH = 5, no-COOH = 1

Please hit RETURN to continue -
```

Text Frame 18: Optimer Grid

(or that it doesn't apply). This process is repeated for each of the elements within the grid. The completed grid can then be listed out (see Text Frame 18). Typing "DONE" will end the grid acquisition process and move on to analyzing the grid for inconsistencies and completeness.

Although analysis of the grid can be accomplished manually, most developers will find the automated analysis and rule generation to be preferable. Analyzing the grid will insure that the generated rulebase will be complete and free from inconsistencies. Once "AUTO" is chosen from the analysis menu, a cluster analysis will be performed (1) first on the elements than (2) on the traits.

The results from performing a cluster analysis on the elements is shown in Text Frame 19. The numbers represent how close a match the two particular elements are. For example, it can be inferred from Text Frame 19 that the grid views the molecules found on pages 5-7 and pages 8-9 (elements 2 and 3) as containing 84% of the same functional groups. Problems will occur in the system generated from the grid when match values exceed 90%.

Below are listed the cluster analysis results of the elements:

	element	1	2	3	4	5	6	7	8	9
1.	p-4	--	75	75	84	59	59	59	50	50
2.	p-5-7		--	84	59	84	67	50	42	42
3.	p-8-9			--	59	67	84	50	42	42
4.	p-10-11				--	75	75	59	50	50
5.	p-12					--	84	50	42	42
6.	p-13						--	50	42	42
7.	p-14							--	75	75
8.	p-15								--	84
9.	p-16									--

Please hit RETURN to continue -

Text Frame 19: Element Cluster Analysis

If there are match values that exceed 90%, then the developer will be prompted concerning methods to resolve the inconsistency.

Following this, the traits will then be analyzed using cluster analysis. The results, shown in Text Frame 20, are analogous to those from the elements. The values shown in the lower left-hand quadrant represent the results from comparing one trait's left-hand pole to the others right-hand pole. This is done because the developer decides on assigning the poles to the traits. Again, if there are match values that exceed 90%, then the developer will be prompted concerning methods to resolve the inconsistency. For additional information on the analysis techniques, see section IV D.

Below are listed the cluster analysis results of the traits:
(Negative values indicate that traits are opposites.)

trait	1	2	3	4	5	6
1. CdO	--	0	34	34	0	12
2. has-benz	-66	--	-22	0	12	0
3. OsH	12	0	--	34	0	12
4. N3	12	0	-11	--	23	34
5. NP	23	-11	23	23	--	67
6. COOH	34	0	12	56	45	--

Note the numbers in the lower left-hand side of the table result from inverting one of the trait poles before doing the analysis.

Please hit RETURN to continue -

Text Frame 20: Trait Cluster Analysis

To this point, all of the traits have been treated equally. However, the developer may view several of the traits as being more important in determining the final answer than others. This is accommodated by rating the importance of each trait on a scale of 1 to 5, 5 being the

most important. In OPTIMER, all of the traits are given equal importance, as shown in Text Frame 21.

```
Please rate each of the following traits based on their importance to the
  problem at hand, using a scale of 1 (least important) to 5 (most important).
  Type any negative number to exit, or "0" to remain unchanged.

The importance of trait (carbonyl) is currently 0, new value is -> 5
The importance of trait (aromatic-ring) is currently 0, new value is -> 5
The importance of trait (hydroxyl) is currently 0, new value is -> 5
The importance of trait (amine) is currently 0, new value is -> 5
The importance of trait (primary-amine) is currently 0, new value is -> 5
The importance of trait (carboxylic-acid) is currently 0, new value is -> 5
```

Text Frame 21: Assigning Trait Weights

This completes the development of the grid. Text Frame 22 shows the completed grid for the OPTIMER system. The automated analysis presents the developer with the rule generation menu, showing the different systems for which rules can be generated. OPTIMER is intended to run with CHESS under VPI Prolog in a VAX environment (see Section IV.E below).

```
4 = p-4
  5-7 = p-5-7
      8-9 = p-8-9
          10- = p-10-11
               12 = p-12
                   13 = p-13
                       14 = p-14
                           15 = p-15
                               16 = p-16
 4 5-7 8-9 10-  12  13  14  15  16 (rating)   trait pairs
5   5   5   1   1   1   3   3   3    (5)   CdO = 5, no-CdO = 1
5   5   5   5   5   5   1   1   1    (5)   has-benz = 5, no-benz = 1
5   1   1   5   1   1   3   3   3    (5)   OsH = 5, no-OsH = 1
3   5   1   3   5   1   3   3   3    (5)   N3 = 5, no-N3 = 1
3   3   3   3   3   3   5   1   1    (5)   NP = 5, no-NP = 1
3   3   3   3   3   3   3   5   1    (5)   COOH = 5, no-COOH = 1
```

Text Frame 22: OPTIMER Grid with Trait Weights

The inference engine written for KEY in VPI Prolog uses four additional pieces of information in executing a rulebase produced by KEY (see Text Frame 23). This information is inserted at appropriate places throughout the dialog to make it a bit more understandable. The first is a

general description of the problem. The second piece of information is a description of the final answer. The third is a description of each trait. The last piece of information is the specific item being considered. In giving a general description of how to generate expert systems using KEY, the term "specific item" must be used because all systems may not be centered around molecules. In the OPTIMER system, the specific item is the target molecule. For example, in the OPTIMER system the general description is "choosing a chiral column", the final answer is "a column", and the traits are "functional groups". Finally, in looking for "functional groups" in trying to decide on "a column", the user is dealing with a specific item, namely a "molecule".

```
Please enter the file name for the rule base.
  Is CHIRAL.HC the correct file name? (*YES or NO) ->
Rule generation for HC Prolog is about to begin.  Note that since there
  are 9 elements and 6 traits, over 108 rules could be generated.
Would you like to proceed?   (*YES or NO) ->

What is a short description of the area you are investigating?
                                  (enter it here) -> choose a chiral column
In looking at choose-a-chiral, what is a description of the final
  answer you are looking for? (enter it here) -> a column
In classifying the different a-columns, what is a description of
  the traits you are using?   (enter it here) -> functional groups
If all of the functional-groups deal with one particular item, what is
  a description of that item? (Return for none) -> molecule

The rule list has been saved in the file CHIRAL.HC.
```

Text Frame 23: Generating HC Prolog Rulebase

The resulting rulebase is saved to a file, and must be transferred to the VAX environment so that the VPI Prolog inference engine developed for KEY can gain access to it. The format of the rulebase generated for OPTIMER is shown in Text Frame 24. The rulebase consists of the additional

information entered above, a list of the traits, and the rules themselves. An English interpretation of the rules is "If the value for the trait carbonyl is that it has a carbonyl group, than there is a 44% chance that the target molecule is found on page 4 of the application guide". With this, the developer completes the development of the OPTIMER rulebase. It is now ready for a user to test out the resulting system.

```
;          ******* File CHIRAL.HC *******

(assert
; generated using PCT

((ques_lst "chiral" "choose-a-chiral" "a-column"
                          "functional-grou" "molecule"
(
     (carbonyl CdO no-CdO)
     (aromatic-ring has-benz no-benz)
     (hydroxyl OsH no-OsH)
     (amine N3 no-N3)
     (primary-amine NP no-NP)
     (carboxylic-acid COOH no-COOH)
           )     ))

((maxrules 108))

((chiral 1 p-4 44) if (fact carbonyl CdO))
((chiral 2 p-4 -44) if (fact carbonyl no-CdO))
((chiral 3 p-5-7 44) if (fact carbonyl CdO))
((chiral 4 p-5-7 -44) if (fact carbonyl no-CdO))
((chiral 5 p-8-9 44) if (fact carbonyl CdO))
((chiral 6 p-8-9 -44) if (fact carbonyl no-CdO))
     .
     .
     .
```

Text Frame 24: OPTIMER Rulebase Format

To run the OPTIMER rulebase, log onto VTCS1 (the Computer Science MicroVAX II), set default to the directory "fgroup.dir", and type "HCC RES" to load the VPI Prolog KEY inference engine. Once it loads, type "run" to start the KEY inference engine and provide the

```
In looking at choose-a-chiral, trying to determine a-column,
     what molecule would you like to investigate?
     Enter the molecule name here --> MANDELIC ACID
  [MANDELIC ACID] currently exists in runtime Molecule database.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-groups.
       (molecule name is ->MANDELIC ACID<- )
```

Text Frame 25: Choosing a Molecule for OPTIMER

name of the KEY rulebase. The inference engine will ask the user for the molecule for which they wish to choose a chiral column (see Text Frame 25), and then run the OPTIMER system.

It is in the running of the OPTIMER rulebase that KEY takes advantage of the hooks built into the CHESS expert system shell. In typical systems constructed using Personal Construct Theory, the user would have to answer a series of questions to determine the value of each trait for the target molecule. However, because CHESS understands what a functional group is and how to determine if it is present within a molecule, these questions are unnecessary (see Text Frame 26). Rather than questioning the user, CHESS determines that there is a carbonyl group present in the molecule Mandelic Acid.

```
Looking for a C=O in MANDELIC ACID:
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
*    Found C=O bond in MANDELIC ACID at atom 9.
*   For carbonyl, Found that CdO exists. [MANDELIC ACID]
```

Text Frame 26: KEY using "Hooks" built into CHESS

If CHESS does not have the hook built into it for a particular functional group query, or if KEY makes a request that CHESS does not understand, the user will be asked to determine if that functional group is present. Text Frame 27 demonstrates this procedure.

```
WHICH OF THE FOLLOWING IS TRUE ABOUT    aromatic-ring
      A] has-benz
      B] no-benz
Please type in your response as A/B ->  A
*  You chose A, which was has-benz.
```

Text Frame 27: KEY Querying User for Trait Value

After answers for all of the traits are obtained either through accessing CHESS's hooks or asking the user, KEY accesses the rulebase and rank orders the list of page numbers where the user should look for molecule/column pairs related to their molecule. The results for running OPTIMER with the molecule Mandelic Acid are shown in Text Frame 28. Tests show that OPTIMER accurately represents the decision tree knowledge contained within the Daicel Chiral Column selection guide (for complete results, see Chapter VIII).

```
We are currently looking at choose-a-chiral, trying to decide
     on which a-column to choose based on functional-groups.
               (molecule name is ->MANDELIC ACID<- )
p-4    WITH A PROBABILITY OF   82.
p-8-9   WITH A PROBABILITY OF   75.
p-10-11   WITH A PROBABILITY OF   55.
p-15   WITH A PROBABILITY OF   55.
p-13   WITH A PROBABILITY OF   35.
p-5-7   WITH A PROBABILITY OF   35.
p-16   WITH A PROBABILITY OF   -16.
p-12   WITH A PROBABILITY OF   -53.
p-14   WITH A PROBABILITY OF   -69.
```

Text Frame 28: OPTIMER Results

Now that OPTIMER has been demonstrated, a further discussion on some of the points not covered during its development is warranted. These sections will go into many of the technical aspects of Personal Construct Theory not previously elaborated upon.

IV.C. Grid Acquisition

Once a topic is chosen and all background information is completed, the KEY system can be used to automatically construct an expert system. To begin a KEY session on any DOS based PC, install the KEY program and then type "KEY6"

(refer to Chapter VII.A for a complete description of installing and running KEY). The first menu encountered has to do with the display size being used (see Text Frame 29). This is used when displaying larger grids on smaller screens, allowing all of the grid to be seen a screen at a time. For most applications, selection #1 will be suitable.

```
Below are listed the display modes that are currently available for use.
        The numbers indicate columns(or characters) by rows(or lines)
        currently being shown by this display:

                1)  80 x 25 - Any Display
                2)  80 x 43 - Enhanced Display
                3) 132 x 25 - Multi Display
                4) 132 x 43 - Multi Display
                5)  80 x 60 - Multi Display
                6) 120 x 25 - Enhanced Display
                7) 120 x 43 - Enhanced Display
                8) 132 x 25 - Color Display

Please enter the number corresponding to the current display size: 2

Your display should display 80 columns and 43 rows.
```

Text Frame 29: KEY Screen Size Selection

The KEY menuing system (see Text Frame 12) presents the developer with all currently valid commands (in capital letters) along with a brief description of what each does. If the command entered is not understood, the current menu will be re-displayed, noting the command entered by the developer. The menuing system also allows the developer to type HELP at any time in order to get a menu showing all of the allowable selections.

The first step in eliciting the knowledge grid is to obtain the elements. If the elements have been previously entered, choose GET from the element menu (see Text Frame 12) and input the file name when prompted to

retrieve the previously saved element list as shown in Text Frame 30. Remember that the elements are simply the answers that the developer wishes the finished system to give. Alternatively, they can be entered in, one at a time (see Text Frame 13). For each element, a name and a three letter abbreviation is required. The abbreviation is used to conserve space when displaying the completed grid. At any time, elements already entered can be listed or edited. Once all of the elements have been entered, they should be saved so they can be recalled at a later time. Choose SAVE (see Text Frame 31) from the menu of choices before going on to the traits, and provide the file name when prompted.

```
Please enter the name of the file that contains the
        element list.

If no extension is entered, .ELM will be assumed (type
        RETURN to exit)

  Filename -> chiral1b

  Is CHIRAL1B.ELM the correct file name? (*YES or NO) ->

 The element list has been restored.
```

Text Frame 30: Getting a Saved Element List

```
Please enter your choice here -> save

Please enter name of the file to contain the saved
        element list.

If no extension is entered, .ELM will be assumed. (type
        RETURN to exit)
    Filename -> chiral
    Is CHIRAL.ELM the correct file name? (*YES or NO) ->

The element list has been saved.
```

Text Frame 31: Saving KEY Element Lists

The information required for each trait, which are the characteristics used to distinguish between each of the elements, is slightly different from the elements. Remembering that the traits are bi-polar, a name of the left

and right pole of the scale is needed, in addition to a general description of the trait. Traits can be a characteristic that the elements either have or do not have, like the trait "CdO/no_CdO" referring to a carbonyl (C=O) functional group (see Text Frame 15). In addition, the two poles of the scale allow for the representation of traits that posses a more continuous quality such as "gentle/rough". This allows the developer to distinguish between a 7 year old that likes to play rough (a 2 on the scale of 1 to 5) and a 14 year old bully that is rough (definitely a 1).

These traits can be read in from a previous session by selecting GET from the menu and supplying a file name (as in Text Frame 30), or they can be elicited in one of two fashions. In addition to querying the developer as described above, KEY has the ability to input a list of traits that has already been determined. This is done in a fashion similar to that of entering the elements, by choosing "EDIT" from the trait menu, then "ADD" to add each trait (see Text Frame 32), answering the questions with the

```
You want to add another trait BEFORE what trait number (enter '0' to
     LIST traits, any negative number to EXIT without additions,
     or any large number to append the trait to the end) -> 8

The name of trait 4 is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) N3
The trait's opposite is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) no N3
The description of trait 4 is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) amine
               The name of trait 4 is {N3}.
                    and its opposite is {no-N3}.
               The description of trait 4 is {amine}.
```

Text Frame 32: Adding to the Trait List

appropriate information. Note that the analysis of the grid will determine if additional traits are needed, and they can then be added at that time. Again, the traits should be saved, so they can be recalled at a later time, by choosing SAVE from the menu and providing a file name (as in Text Frame 31) before proceeding to developing the grid.

The remaining task is to fill in the grid with numbers that represent each elements value with respect to each trait. Here again, a grid that had been saved previously can be read in by selecting GET from the menu and typing the file name at the prompt. Entering a new grid has already been covered (see Text Frame 17), and refer to the discussion of the work of Gains and Shaw on page 55 for additional background explanation of grid values. Using the trait rough/gentle, the grid values can be interpreted as follows: 5 means the element can be considered gentle (the traits left-hand pole), 4 is sort of gentle, 2 is a little rough, 1 is rough (the traits right-hand pole), and 3 means the trait doesn't apply to the element. In systems such as OPTIMER where the traits represent questions such as having or not having a carbonyl group, values of 2 or 4 do not have a meaning.

This will result in a completed Grid, which should be saved as before. In order to insure that the list of traits

is sufficient to fully distinguish between each of the elements, some analysis of the grid is required.

IV.D. Grid Analysis

OPTIMER's goal is to direct the user to the correct location within the Daicel Applications Guide to find molecules similar to the user's sample. In doing this, PCT ascertains the value for each of the traits with respect to the user's sample, then determines which of the elements is the correct response based on the information contained within the grid. It is the analysis of the knowledge grid that insures a system generated using KEY will be complete and free of inconsistencies.

The analysis menu presents the various choices the developer has for insuring the accuracy of the finished system. In addition to the analysis of the grid, there are a number of other functions handled in this menu (see Text Frame 33). The match value is the threshold above which the analysis will report problems found within the grid. Most researchers utilize a value of 90%, indicating that there is a problem if two traits or

```
Your choices for analyzing the present grid are as follows:

There are 10 commands you can enter, and they are:
    CH-MATCH - change the value at which matches are detected
    E-ANAL - analyze the elements for similarities
    T-ANAL - analyze the traits for similarities
    RATE-T - determine the importance of each trait
    RULES - generate rules from the current grid
    AUTO - automatically do the analysis and rule generation
    SAVE-ALL - save the elements, traits, and grid.
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> ch-match
```

Text Frame 33: Grid Analysis Menu

elements are more than 90% similar. This value can be changed from its 90% default value for both automated and manual grid analysis by choosing "CH-MATCH" from the analysis menu.

Although KEY can will go through the analysis of the grid automatically (as covered in section IV B), developers may wish to manually analyze a grid that contains a lot of inconsistencies once they gain some familiarity with the process. The analysis menu indicates that both the elements and traits need to be analyzed (see Text Frame 33). Choosing "E-ANAL" will analyze the elements for inconsistencies and completeness. The menu for analyzing and editing the elements lists the various techniques available for testing the elements, along with editing and listing options (see Text Frame 34).

```
Your choices for editing the grid elements are as follows:

There are 9 commands you can enter, and they are:
   E-CORR - perform correlation analysis on the elements
   E-CLUST - perform cluster analysis on the elements
   RPT-EMAT - report matches found while analyzing the elements
   EDIT-E - edit the list of elements
   EDIT-T - edit the list of traits
   LIST-E - list out the elements
   LIST-G - list out the grid
   DONE - finished entering value list
   HELP - print out this list of commands

Please enter your choice here -> e-corr
```

Text Frame 34: Element Analysis Menu

The analysis techniques available are correlation and cluster analysis. If one were to plot the grid data in an "n" dimensional space with regards to both the elements and the grids, both techniques define a distance measure between

the two items being compared. They differ in the way they measure the distance and in the way they report the results. These techniques will allow the developer to see what elements (or traits) appear to be redundant, indicating either that one can be discarded or that additional information is required. This phase actually consists of two parts: the first involves the actual mathematical analysis of the grid, and the second involves the interpretation of the results.

The correlation analysis of the elements is shown in Text Frame 35. The numbers represent the similarity between the elements on a scale that is based on the number of traits used. The interpretation of the numbers is intuitive, i.e. the higher the number the more alike they are. Note that the largest possible number would be two times the number of traits.

Below are listed the correlations between the elements:

	element	1	2	3	4	5	6	7	8	9
1.	P4	--	8	8	10	6	6	2	2	2
2.	P5-7		--	10	6	10	8	4	0	0
3.	P8-9			--	6	8	10	2	0	0
4.	P10-11				--	8	8	2	2	2
5.	P12					--	10	4	0	0
6.	P13						--	2	0	0
7.	P14							--	6	6
8.	P15								--	10
9.	P16									--

The maximum correlation between any two elements is 12.

Text Frame 35: Element Correlation Analysis

Calculating the correlation matrix involves comparing the two elements for each trait within the grid. If the two grid values match exactly, two is added to the correlation sum. If there is a partial match, one is added. A partial

match is defined as both objects showing belief in the same side of the construct (i.e. the two grid values would be a 5 & 4 or a 1 & 2). In order to use this measure of similarity in the automated analysis of the grid, the correlation sum is best converted to a scale of zero to one hundred. This value is calculated according to Equation (2):

$$C = \frac{S * 100}{N * 2} \qquad (2)$$

where C is the scaled correlation, S is the correlation sum arrived at by summing the individual correlations as noted above, and N is the number of traits or elements being analyzed.

Cluster analysis of the grid elements involves measuring the distance between the two elements (see Text Frame 19). Remember that the grid values are on a scale of one to five, and that they represent beliefs in the constructs: a value of five represents belief in the left-hand pole, whereas a value of one represents belief in the right-hand pole. The cluster score for the elements is shown in Equation (3):

$$S = \frac{-100 * d_{ij}}{\Delta V * N} \qquad (3)$$

where S is the cluster score, d_{ij} is the difference measure between elements i and j, δV is the largest possible

difference between any two grid values (i.e. 5 - 1 = 4), and N is the total number of elements. The difference measure is arrived at by subtracting one elements grid value from the others, taking the absolute value, and summing all the differences from each trait. The cluster score for the elements ranges between 0 and 100. Elements are compared directly, resulting in only half of their matrix being completed (see Text Frame 19). The automated analysis of the grid uses cluster analysis which many researchers in the literature report for their data[73].

A list of pairs of elements that have been found to be closely related is compiled as a result of the cluster analysis. The developer should continue to refine the elements and reanalyze until no more matches are discovered. Once the elements have been analyzed, the traits should be examined.

Choosing "T-ANAL" from the analysis menu (see Text Frame 33) will analyze the traits for inconsistencies and completeness. The menu for analyzing and editing the traits is analogous to that of the elements, showing the various techniques available for testing

```
Your choices for editing the grid traits are as follows:

There are 9 commands you can enter, and they are:
    T-CORR - perform correlation analysis on the traits
    T-CLUST - perform cluster analysis on the traits
    RPT-TMAT - report matches found while analyzing the traits
    EDIT-T - edit the list of traits
    EDIT-E - edit the list of elements
    LIST-T - list out the traits
    LIST-G - list out the grid
    DONE - finished entering value list
    HELP - print out this list of commands


Please enter your choice here ->
```

Text Frame 36: Trait Analysis Menu

the traits, along with the same editing and listing options (see Text Frame 36).

Calculating the correlation matrix for the traits involves comparing the two traits for each elements within the grid (see Text Frame 37). It is done in a similar fashion as the with the elements (see above). The numbers represent the similarity between the traits on a scale based on the number of elements contained within the grid. The meaning of the numbers is as before: the higher the number the more alike they are. The upper limit would be two times the number of elements.

Below are listed the correlations between the traits:

	trait	1	2	3	4	5	6
1.	Benz	--	6	4	4	4	2
2.	CdO		--	12	8	0	2
3.	OsH			--	8	0	2
4.	N3				--	6	4
5.	NP					--	14
6.	COOH						--

The maximum correlation between any two traits is 18.

Text Frame 37: Trait Correlation Analysis

The cluster analysis results of the traits is shown in Text Frame 20. Calculating the cluster score for the traits is slightly different, as shown in Equation (4):

$$S = \frac{-200 * d_{ij}}{\Delta V * N} \qquad \textbf{(4)}$$

where S, d_{ij}, δV, N, and the difference measure are defined as in Equation (3) above except they represent is the total number of elements. Traits can be compared in both the forward and reverse fashions because of their bi-polar

nature, resulting in the entire matrix being completed. The cluster score for the traits ranges between -100 and 100. A score of 100 means that the two are exactly alike, while a score of zero means they are not alike at all. A score of negative one hundred means that the traits are exact opposites of one another.

Most researchers utilize either cluster or correlation analysis. However, there is no standard regarding how they use the results of these analysis to point out inconsistencies, repetitiveness, or flaws present in the data, and then how they use this information to implement corrective procedures. Cluster analysis was chosen over correlation analysis for use in the automatic analysis of the grid because of the additional scaling required for correlation analysis.

In an effort to demonstrate the grid editing procedures a developer would use in modifying the grid after it was developed, an additional element will be added to the OPTIMER system. It was discovered that molecules on page 17 were inadvertently omitted from the final system. Choosing

```
Your choices for editing the element list are as follows:

There are 7 commands you can enter, and they are:
    CHANGE - change one element in the list of elements
    ADD - add one element to the list of elements
    REMOVE - remove one element from the list of elements
    LIST - list out the elements
    LIST-G - list out the grid
    DONE - finished entering element list
    HELP - print out this list of commands

Please enter your choice here -> add
```

Text Frame 38: Grid Element Edit Menu

"EDIT-E" from the analysis menu (see Text Frame 33) presents the developer with the element editing menu, shown in Text Frame 38.

The process of adding an element to the grid begins with choosing "ADD" from the element editing menu (see Text Frame 38). After choosing the location within the element list where the developer wants to position the new element (usually at the end of the list), its name and three letter abbreviation are entered (see Text Frame 39). Following this, KEY prompts the developer to input values for the element for each of the traits currently within the knowledge grid.

```
You want to add another element BEFORE what element number
      (enter '0' to LIST elements, any negative number to EXIT
      without additions, or any large number to append the
      element to the end) -> 10

The name of element 10 is currently {}.  What would you like it
      to be? (RETURN to remain unchanged) Page number 17

The element's abbreviation is currently {}.  What would you
      like it to be? (RETURN to remain unchanged) P17

      The name of element 10 is {Page number 17},
        and its abbreviation is {P17}.

Please rate the element {Page number 17} on each of the
      following traits using a scale of 1 to 5. (type a
      negative number to exit, 0 to remain unchanged)

Trait {Benz} is 5 and {No-Benz} is 1.
      Its current value is 0, the new value is -> 5
Trait {CdO} is 5 and {No-CdO} is 1.
      Its current value is 0, the new value is -> 1
Trait {OsH} is 5 and {No-OH} is 1.
      Its current value is 0, the new value is -> 1
Trait {N3} is 5 and {No-N3} is 1.
      Its current value is 0, the new value is -> 1
Trait {NP} is 5 and {No-NP} is 1.
      Its current value is 0, the new value is -> 5
Trait {COOH} is 5 and {No-CO2H} is 1.
      Its current value is 0, the new value is -> 3
```

Text Frame 39: Adding Elements to the Grid

At any time during the grid acquisition process, the developer can edit both the elements and the traits. To change an element currently in the element, choose "CHANGE" from the element editing menu (see Text Frame 38). This

begins a dialog, shown in Text Frame 40, where the developer is asked to verify first the element name and then it's extension. Then the developer is presented with and given the opportunity to change the current value for the element for each trait.

```
Which element number do you want to change? (enter '0' to LIST
        elements, any negative number to EXIT without making
        changes) -> 10

The name of element 10 is currently {Page number 17}.  What
        would you like it to be? (RETURN to remain
        unchanged) -> Page 17

The element's abbreviation is currently {p17}.  What would you
        like it to be? (RETURN to remain unchanged) -> P17

           The name of element 10 is {Page-17}.
                and its abbreviation is {P17}.

Please rate the element {Page-17} on each of the following
        traits using a scale of 1 to 5. (type a negative
        number to exit, 0 to remain unchanged)

 Trait {Benz} is 5 and {No-Benz} is 1.
        Its current value is 5, the new value is -> 0
 Trait {CdO} is 5 and {No-CdO} is 1.
        Its current value is 1, the new value is -> 0
 Trait {OsH} is 5 and {No-OH} is 1.
        Its current value is 1, the new value is -> 0
 Trait {N3} is 5 and {No-N3} is 1.
        Its current value is 1, the new value is -> 0
 Trait {NP} is 5 and {No-NP} is 1.
        Its current value is 5, the new value is -> 0
 Trait {COOH} is 5 and {No-CO2H} is 1.
        Its current value is 3, the new value is -> 0
```

Text Frame 40: Changing a Grid Element

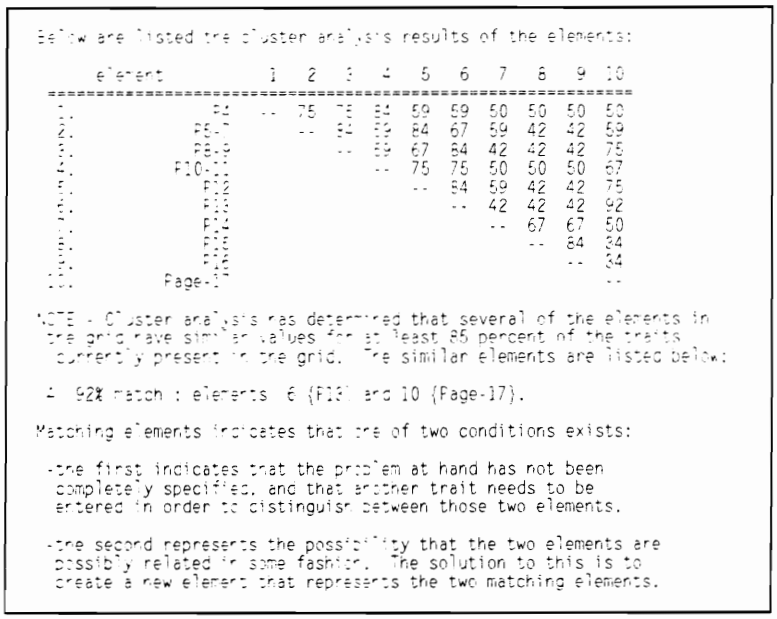
The newly added element is now part of the knowledge grid, as shown in Text Frame 41. Now that changes have been made to the grid, the grid analysis must be performed again to insure that the grid is complete and free of inconsistencies.

```
P4  = P4
    P5  = P5-7
        P8  = P8-9
            P10 = P10-11
                P12 = P12
                    P13 = P13
                        P14 = P14
                            P15 = P15
                                P16 = P16
                                    P17 = Page-17
 P4  P5  P8  P10 P12 P13 P14 P15 P16 P17       trait pairs
  5   5   5   5   5   5   1   1   1   5   Benz = 5, No-Benz = 1
  5   5   5   1   1   1   3   3   3   1   CdO = 5, No-CdO = 1
  5   1   1   5   1   1   3   3   3   1   OsH = 5, No-OH = 1
  3   5   1   3   5   1   5   3   3   1   N3 = 5, No-N3 = 1
  3   3   3   3   3   3   5   1   1   5   NP = 5, No-NP = 1
  3   3   3   3   3   3   3   5   1   3   COOH = 5, No-CO2H = 1
```

Text Frame 41: Changed OPTIMER Grid

The analysis of the elements is performed as described previously in Text Frame 33 and Text Frame 34. The results of the cluster are in a format similar to that described for

Text Frame 19. This time the cluster analysis of the elements shown in reveals that there is a 92% match between the element just added (Page 17) and element 13 (P 13). This match, along with any others that are found, are noted and possible corrective actions suggested. Because this analysis was done manually, it is left up to the developer to resolve them by editing the grid manually.



Below are listed the cluster analysis results of the elements:

	element	1	2	3	4	5	6	7	8	9	10
1.	P1	--	75	75	84	59	59	50	50	50	50
2.	P5-7		--	84	59	84	67	59	42	42	59
3.	P8-9			--	59	67	84	42	42	42	75
4.	P10-11				--	75	75	50	50	50	67
5.	P12					--	84	59	42	42	75
6.	P13						--	42	42	42	92
7.	P14							--	67	67	50
8.	P15								--	84	84
9.	P16									--	84
10.	Page-17										--

NOTE - Cluster analysis has determined that several of the elements in the grid have similar values for at least 85 percent of the traits currently present in the grid. The similar elements are listed below:

4 - 92% match : elements 6 {P13} and 10 {Page-17}.

Matching elements indicates that one of two conditions exists:

-the first indicates that the problem at hand has not been completely specified, and that another trait needs to be entered in order to distinguish between those two elements.

-the second represents the possibility that the two elements are possibly related in some fashion. The solution to this is to create a new element that represents the two matching elements.

Text Frame 42: Element Cluster Analysis Showing Inconsistency

The automated grid analysis proceeds slightly differently. Choosing "AUTO" from the grid analysis menu shown in Text Frame 33 will start the automated grid analysis. After a brief explanation of the automated analysis procedure as shown in Text Frame 43, a cluster analysis of the elements is performed. Here the cluster analysis uncovers a 92% similarity between the elements {P13} and {Page 17}.

This similarity will result in the rulebase not being able to distinguish between these two elements in trying to

suggest where to go to look for a molecule similar to a user's target molecule. The developer is presented with the menu of choices shown in Text Frame 44 to help to resolve the similarity. Selecting "GO-ON" will allow the developer to resolve another similarity in the match list, which can be displayed by selecting "RPT-EMAT". The mechanism to "INPUT" another trait to further distinguish between the elements is analogous to that for adding an element shown in Text Frame 39. The developer can also verify that the values in the grid are correct for the two elements, or combine them into a single element.

The procedure for automated analysis is about to begin. This procedure will take you through the analysis of both the element and traits to resolve many inconsistencies found to be present. Following this procedure, options will be presented to allow generation of a rule-based system reflecting the information contained in the grid.

Below are listed the cluster analysis results of the elements:

	element	1	2	3	4	5	6	7	8	9	10
1.	P4	--	75	75	84	59	59	50	50	50	50
2.	P5-7		--	84	59	84	67	59	42	42	59
3.	P8-9			--	59	67	84	42	42	42	75
4.	P10-11				--	75	75	50	50	50	67
5.	P12					--	84	59	42	42	75
6.	P13						--	42	42	42	92
7.	P14							--	67	67	50
8.	P15								--	84	34
9.	P16									--	34
10.	Page-17										--

It has been determined that the two elements {P13} and {Page-17} are 92 percent alike. In order to resolve this similarity, you can COMBINE them into a new element, or INPUT another trait to further distinguish between them.

Text Frame 43: Automated Analysis Uncovering Match

It has been determined that the two elements {P13} and {Page-17} are 92 percent alike. In order to resolve this similarity, you can COMBINE them into a new element, or INPUT another trait to further distinguish between them.

There are 9 commands you can enter, and they are:
- COMBINE - combine a pair of element that are closely matched
- INPUT - input a new trait to better distinguish between two elements
- VERIFY - verify all the grid values for the two matching elements
- GO-ON - consider the next pair of matching elements
- FINISHED - finished with the analysis of the elements
- RPT-EMAT - report matches found while analyzing the elements
- LIST-G - list out the grid
- QUIT - exit the automated analysis and generation process
- HELP - print out this list of commands

Please enter your choice here ->

Text Frame 44: Similarity Resolution Menu

The first step in resolving a similarity is to insure that the data entered into the grid is accurate. After selecting "VERIFY" from the menu, the developer is presented with the two similar elements and asked to make sure that each grid value is accurate. The two elements are presented individually as shown in Text Frame 45.

```
Please verify the current data for the element {P13}:

Please rate the element {P13} on each of the following
       traits using a scale of 1 to 5. (type a
       negative number to exit, 0 to remain unchanged)

Trait {Benz} is 5 and {No-Benz} is 1.
       Its current value is 5, the new value is -> 0
Trait {CdO} is 5 and {No-CdO} is 1.
       Its current value is 1, the new value is -> 0
Trait {OsH} is 5 and {No-OH} is 1.
       Its current value is 1, the new value is -> 0
Trait {N3} is 5 and {No-N3} is 1.
       Its current value is 1, the new value is -> 0
Trait {NP} is 5 and {No-NP} is 1.
       Its current value is 3, the new value is -> 0
Trait {COOH} is 5 and {No-CO2H} is 1.
       Its current value is 3, the new value is -> 0
```

Text Frame 45: Verifying an Element

If the developer determines that the two elements do in fact represent the same information, they can be combined into a single element. After displaying the names and abbreviations of the two elements and verifying they are the correct two to be combined, the developer is asked to enter a new element name and abbreviation (see

```
The two elements that are 92 percent alike are listed below :
          NAME :   P13                  Page-17
    ABBREVIATION:   P13                  P17
These two elements are to be combine? (YES* or NO)

What is the name for the new element? P13
     What is the abbreviation for {P13}? P13

Now rate {P13} on each of the following trait.  The current
       values of the two elements that are being combine are
       shown on the left-hand side of each line.

 P13 = P13
   P17 = Page-17              Current trait              P13
=============================================================
   5  5 - Value for Aromatic [Benz=5, No-Benz=1] ->        5
   1  1 - Value for Carbonyl [CdO=5, No-CdO=1] ->          1
   1  1 - Value for Alcohol [OsH=5, No-OH=1] ->           1
   1  1 - Value for Tert-Nitrogen [N3=5, No-N3=1] ->      1
   3  5 - Value for Primary-Amine [NP=5, No-NP=1] ->      3
   3  3 - Value for Carboxylic-Acid [COOH=5, No-CO2H=1] -> 3
```

Text Frame 46: Combining Two Elements

Text Frame 46). Then the grid data for both of the elements is presented and the developer is asked to enter the grid data for the new element.

After each change or addition is made to the grid, the automated analysis procedure will perform another cluster analysis of the elements. As long as further similarities are discovered, the editing process should be repeated. As soon as the automated analysis procedure does not detect any similarities, it will proceed directly to analyze the traits. The automated analysis of the traits occurs in much the same fashion as with the elements, again proceeding only after all similarities have been resolved. If the developer chooses not to resolve all of the similarities uncovered in the grid, selecting "FINISHED" from the menu shown in Text Frame 44 will direct the automated analysis procedure to ignore the remaining similarities and proceed to analyzing the traits. This is only suggested if the developer has set a match level below the recommended 90%.

This section has described the method by which all of the defects are removed from the grid. The analysis can be done manually by the developer: performing the cluster analysis, looking for match values exceeding 90% as reported by the cluster analysis, and editing the grid to remove the similarity. This process is an iterative one, with the developer performing the analysis, removing the defect, then

re-doing the analysis to see if additional similarities exist. An alternative approach, and the one often taken by new developers, is to have the analysis done automatically on first the elements, then the traits. In either case, similar elements indicate that either the answers can be combined into one element, or that an additional trait needs to be specified that distinguishes between them. Traits that are reported as similar usually represent the same type of information (remembering from Chapter II loving/mean and friendly/unfriendly) and can be combined into one trait.

Once the analysis is completed, the final task before rules can be generated from the grid is to rate each of the traits. This allows the developer to treat some traits as more important in arriving at an answer than others. It can be accomplished by choosing "RATE-T" from the analysis menu (see Text Frame 33) if the analysis is being done manually. Personal Construct Theory allows each trait to be given an individual importance, on a scale of one to five, with five referring to the most important trait and one the least. The mechanism is the same as when it was done automatically during the automatic grid analysis (see Text Frame 21).

IV.E. Knowledge Base Generation

The final phase involves the generation of rules making up the knowledge base. The collection of rules making up a

knowledge base is often called a rulebase. Completing either the automated or manual analysis of the grid will result in displaying the rule generation menu shown in . It should be emphasized that any rulebase generated from a grid that has not been thoroughly analyzed will result in a system that will not accurately represent the problem the developer is attempting to model.

```
Your choices for generating rules from the present grid
       are as follows:

There are 6 commands you can enter, and they are:
    T-PROLOG - generate a turbo-prolog system
    HC-PROLOG - generate an hc-prolog system
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> done
```

Text Frame 47: Rule Generation Menu

The argument could be put forth that various properties could be included in the database, and that when queried, if the specific molecule asked for was not found, the system would return a molecule that had properties within a certain range of the molecule asked for. Information as to what constitutes a similar molecule and how to determine the similarity is incorporated into the database. In effect, the combination of such knowledge with a database results in the generation of a knowledge base. A database normally only contains facts. A knowledge base contains not only facts, but also rules stating how those facts are interrelated.

The actual format of the rules and how the inference engine operates are interrelated. KEY was designed to

generate rules for two different inference engines: one running under Turbo-Prolog on a DOS based PC, and the other running under VPI Prolog on a VAX-VMS machine. Both inference engines serve similar tasks, that of executing the rulebase generated by the KEY system, and are shown in Appendix E. The VPI Prolog inference engine, however, has hooks built in to allow it to take advantage of the general chemical knowledge within CHESS, such as molecular representations and functional group recognition.

The format of the rulebase generated by KEY requires some explanation (see Text Frame 24 and Appendix B). There are four different sections to this rulebase: The first represents the additional information required by the VPI Prolog Inference engine (see Text Frame 23), the second is information on each trait, the third is the maximum number of rules generated by KEY, and last is the list of rules generated from the grid using Personal Construct Theory. The individual trait information consists of a description of the trait, followed by the traits left-hand and right-hand poles. The maximum number of rules is determined by doubling the product of the number of elements times the number of traits, since there are two rules generated to describe each traits value for each element.

The rule generation procedure begins with the first element, then looks at its value for each of the traits. An

individual rule is generated for each pairing of the elements with the two trait poles (see Text Frame 24). The implication strength (I.S.) is a numerical value that represents the amount of belief placed in each rule and is calculated as shown in Equation (5) for the right-hand pole:

$$I.S. = T * G * S \tag{5}$$

where T is the trait factor, G is the grid factor, and S scales the implication strength according to the number of traits. The trait factor, T, is twice the trait weight (i.e. 2, 4, 6, 8, or 10). The grid factor indicates the belief strength found in the grid[112]: for a grid value of 1 or 5, G = 10, for a grid value of 2 or 4, G = 5, and for a grid value of 3, G = 0. The scaling factor, S, is calculated based on the number of traits, and is a result of the algorithm that will be used to combine the certainty factors in the finished rulebase[113]. This algorithm will be discussed in the following section on the inference engine. The implication strength for the left-hand pole is the logical negation of that of the right-hand pole. A rule will not be generated if there is a grid value of "3", indicating there is no preference for that particular element with respect to the trait.

One final note that researchers utilizing PCT[77,85] point out. Equation (5) defines the implication strength as a

product of a trait factor, a grid factor, and a scaling factor. The grid factor depends on the value found in the grid. If the grid value of 3 means that the trait does not apply to that particular element, and results in a grid factor of 0. Elements should not have grid values of 3 for a majority of the traits. Ignoring this can result in a rulebase that gives erroneous results, since there are no rules generated for these cases (i.e. the implication strength is 0 since the grid factor is 0). The methodologies behind cluster analysis and correlation analysis do not check for this condition.

```
?-
?- run
What is the name of the rule base you want to use?
      Please enter it here -> chiral-b

The Rulebase ->chiral-b<- has been loaded.
```

Text Frame 48: Running a KEY Rulebase

The VPI Prolog inference engine (see Appendix E) goes through four stages in executing a rulebase generated by KEY: loading the rulebase, determining the trait's values, searching all of the rules, and finally printing out all of the results. To run the inference engine, type "run" from within VPI Prolog (see Chapter VII.D.2 for complete instructions). The user will be asked for the name of the rulebase, which will then be loaded (see Text Frame 48). If the rulebase indicates that there is an specific item to be dealt with while running the rulebase (the target molecule

for the OPTIMER system), the user will be asked for the item name (see Text Frame 25).

Each of the traits are a bi-polar construct that are used to distinguish between the elements. KEY must query the user as to which end of the construct most accurately describes the target molecule (the specific item) for each of the traits. For example, the first trait in OPTIMER has to do with a carbonyl group being present. KEY needs to determine if "CdO" or "no-CdO" describes the target molecule (meaning if the molecule has or does not have a carbonyl group). This can occur either by taking advantage of the "hooks" built into CHESS (as in Text Frame 26) or by querying the user directly for a response (as in Text Frame 27).

After responses for each of the traits is obtained, KEY begins searching through all the rules to determine which of the elements is appropriate to suggest as the "best" response. Each element will have a certainty factor associated with it that is asserted as a Prolog fact. The higher the certainty factor, the more likely it is the correct response. In looking at each rule, if the current rule is the first one describing an element, the certainty factor for the rule becomes the certainty factor for the element. Certainty factors for successive rules are combined according to Equation (6):

$$NewCF = OldCF + \left(\frac{((100 - OldCF) * CF)}{100} \right) \quad (6)$$

where CF is the certainty factor of the current rule, OldCF is the original certainty factor for the current element, and NewCF is the new certainty factor for the current element. See the predicate "compute_cf ?Ele ?Cf" in module KEY.HC in Appendix E. The scale factor defined in Equation (5) are generated such that combining certainty factors in Equation (6) will result in a perfect score for an element where the target molecule has the same value for each of the traits and has no grid values of 3.

The final task is to print out a list of the elements (page numbers for OPTIMER), ordered by decreasing certainty factors. This is accomplished by doing a simple bubble sort on all of the element certainty factors (see "printresults" in module KEY.HC in Appendix E). When the largest certainty factor is found, it is retracted (or removed from the system), printed out (see Text Frame 28), and the procedure calls itself to find the next largest certainty factor. Once

```
The following is the data for
        (choose-a-chiral : A A A B B A).
    For carbonyl, you chose CdO.
    For aromatic-ring, you chose has-benz.
    For hydroxyl, you chose OsH.
    For amine, you chose no-N3.
    For primary-amine, you chose no-NP.
    For carboxylic-acid, you chose COOH.
```

Text Frame 49: OPTIMER Trait Data

there are no more elements to print out, the responses to each of the traits are printed out and retracted as shown in Text Frame 49.

The Turbo-Prolog inference engine operates in a similar fashion. One of the major differences between Turbo-Prolog and VPI Prolog is Turbo-Prolog's need to have all variables and predicate names defined. This would make it much more difficult to implement CHESS using this Prolog implementation. In fact, Turbo-Prolog can be though of as a typical procedural language that uses some Prolog programming techniques. However, because the inference engine cannot take advantage of the hooks built into CHESS as it is currently written, the user is queried for answers to each of the trait questions queried as shown in Text Frame 27. Refer to Chapter VI.A.5 for a more thorough discussion of the Turbo-Prolog inference engine, and to Appendix E for a listing of the Prolog code.

IV.F. Data Structures and Files

There are three basic data structures used within KEY: one for the elements, one for the traits with their grid values if present (both shown in Figure 8), and one that contains the list of matches found while performing the various analysis techniques (shown in Figure 9). Each is made up of an identification structure which then points to

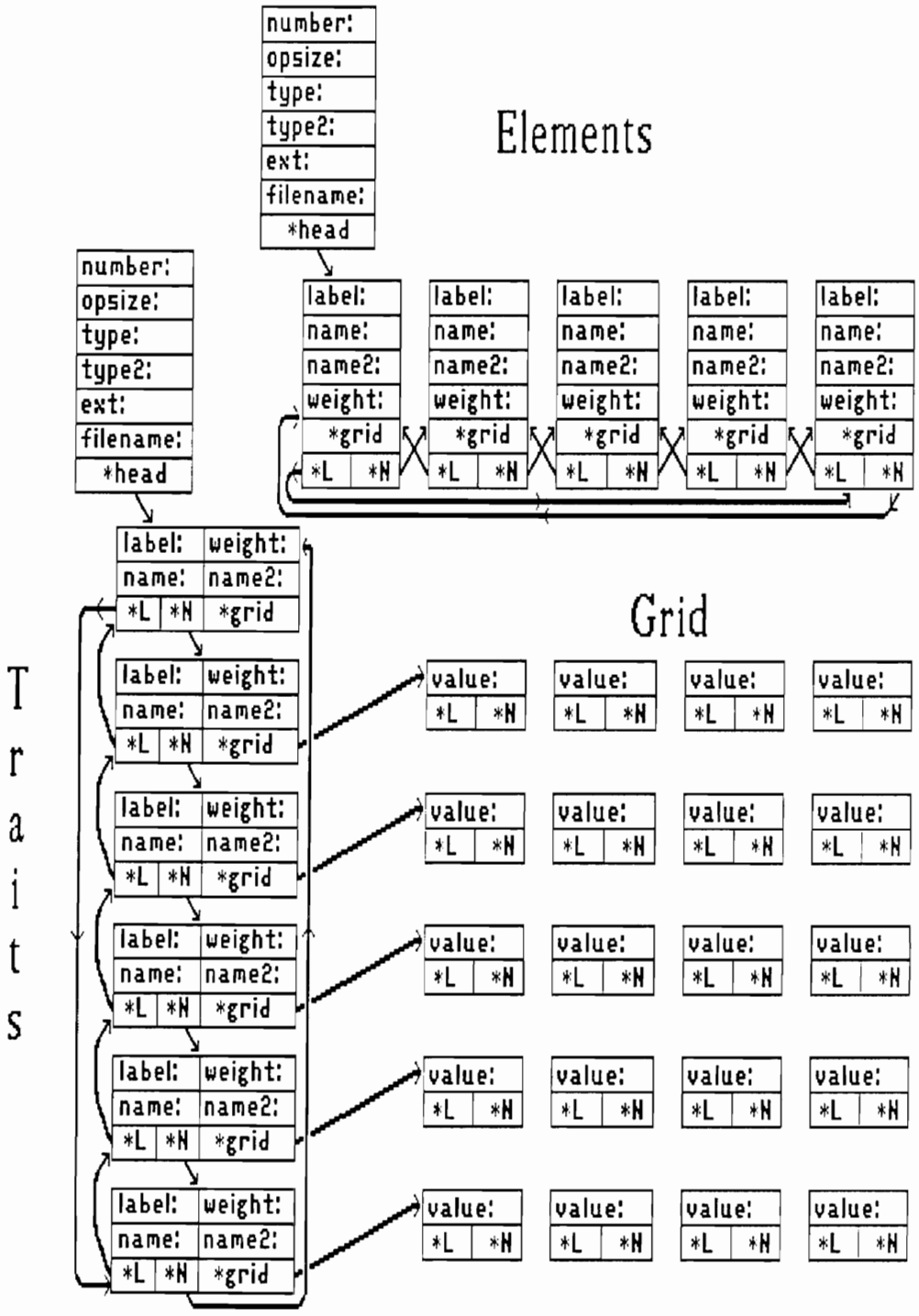


Figure 8: Grid Data Structure

Match Lists

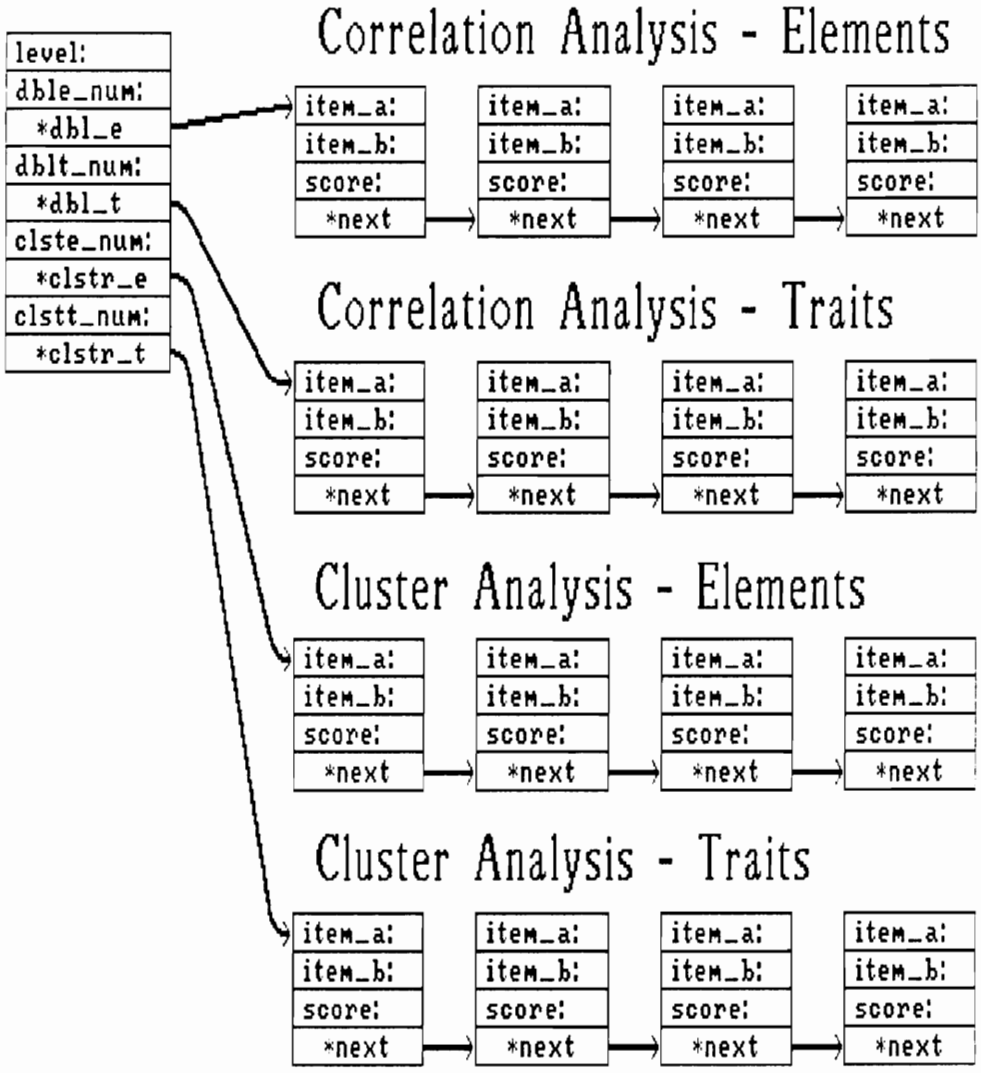


Figure 9: Match List Data Structure

a doubly linked list that contains all of the data. A singly linked list was not used since it would require the program to re-traverse the structure each time the structure on either side was required. Using a doubly linked list enables the program to traverse the structure in both directions, allowing for easy addition and editing of elements and traits. Each of these structures are created dynamically as the elements and traits are entered, when the data is read in from a file, or when the matches are found during the analysis. Each of the structures needs be examined in more detail.

The identification structure for the element list (see Text Frame 50 for "_item" structure definition or Appendix C for the complete definitions) contains the number of elements in "number", the word "element" in "type", the word "abbreviation" in "type2", and "ELM" in "ext". The first two of these are used when printing instructions to the developer, and the third is the default extension used when saving files. The "opsize" field is not used in the element list, and once the element list has been saved, the filename used is stored

```
typedef struct _item {
  int number;
  int opsize;
  char type[9], type2[15], ext[5];
  char filename[14];
  struct _title *head;
  } ITEM;

typedef struct _title {
  char label[NAME_LEN];
  int weight;
  char name[NAME_LEN];
  char name2[NAME_LEN];
  struct _cell *grid;
  struct _title *next, *last;
  } TITLE;

typedef struct _cell {
  int value;
  struct _cell *next, *last;
  } CELL;
```

Text Frame 50: Grid Data Structure

in "filename". The pointer "*head" points to a doubly linked list of "_title" structures, one for each element. There is one extra "_title" structure (see Text Frame 50 and Figure 8) that is used to initialize the doubly linked list, and it uses only its "*next" and "*last" pointers. In all of the "_title" structures, the name of the element is contained in "name" and the 3 letter abbreviation in "name2". The element list does not use "label", "weight", or "*grid".

The structure definition for the trait list is the same as the element list described above, only some of the interpretations change (refer to Text Frame 50 and Appendix C). It contains the number of traits in "number", the word "trait" in "type", the word "opposite" in "type2", and TRT in "ext". Again, these are used when printing instructions to the developer. The "opsize" field is not used in the trait list, and once the trait list has been saved, the filename used is stored in "filename". The pointer "*head" points to a doubly linked list of "_title" structures, one for each element. Again, there is one extra "_title" structure (see Text Frame 50 and Figure 8) that is used to initialize the doubly linked list using only the "*next" and "*last" pointers. In all of the "_title" structures, the name of the right pole of the trait is contained in "name" and the left pole name in "name2". The "label" filed

contains the description of the trait, and "weight" and "*grid" are not used until the grid is added to the trait list.

When the grid is input, it is added to the trait list data structure by using the "*grid" pointer in the structure "_title" (refer to Text Frame 50 and Appendix C). Adding the grid involves creating a new "_cell" structure for each grid value. When the first one is created, the "*grid" pointer is set to it. As new grid values are added, additional "_cell" structures are added into the doubly linked list. Finally, when the rulebase is about to be generated, each trait is assigned a weight which is stored in "weight".

The same structure definitions are used for both the traits and elements so that they could take advantage of the same editing routines. Also, only a singly linked list was required for the grid values since they would only be accessed in order, one after the other.

The match list (see Figure 9) is made up of a master structure which points to each of the individual match lists, one each for both elements and traits for two analysis types (correlation analysis and cluster analysis). The structure definitions are shown in Text Frame 51. The master structure contains the level at which match are to be added to the match list (stored in "level"), and a counter

and pointer to the match list for each of the analysis types. During the analysis, when the comparison of pairs of elements or traits yields a match exceeding the match value stored in "level", a structure "_double" is created with the score put in "score" and the numbers of the elements or traits in "item_a" and "item_b". The new structures are added in such a way that the resulting linked list is sorted, with the highest match occurring first.

```
typedef struct _match {
  int level;
  int dble_num, dblt_num;
  struct _double *dbl_e, *dbl_t;
  int clstre_num, clstrt_num;
  struct _double *clstr_e, *clstr_t;
  } MATCH;

typedef struct _double {
  int item_a, item_b;
  int score;
  struct _double *next;
  } DOUBLE;
```

Text Frame 51: Match List Data Structure

Saving the elements, traits, or grids involves choosing the appropriate selection from the menu and supplying the name for the file when prompted. The files are given the extensions of ".ELM", ".TRT", and ".GRD" respectively, by default. The format of each of the files is similar: The first line contains two numbers, the second line contains three descriptions, and the remaining lines contain the data (see Text Frame 52). For element files, the first number on the first line represents the number of elements, and the second number is zero. The first two descriptions on line two are "element" and "abbreviation", which describe the contents of

```
9 0
element abbreviation .ELM
P4              P4
P5-7            P5
P8-9            P8
P10-11          P10
P12             P12
P13             P13
P14             P14
P15             P15
P16             P16
```

Text Frame 52: KEY Element File

the two columns of information that follows. The last description, ".ELM", is the default extension for the file. The data consists of the name of the element and its 3 letter abbreviation that is used when displaying the completed grid.

Trait files are similarly defined (see Text Frame 53), with descriptions of the data columns being "traits" and "opposites". The data has an additional column, which is the description of the trait. Grid files (also see Text Frame 53) actually contain all of the traits along with the grid information. The first number represents the number of trait - opposite - description data triads are contained within the file, while the second number refers to the number of elements used to create the grid. When loading a grid, this number will be compared with the number of elements currently within KEY. If they do not agree, KEY will display an error message. There is no way to insure that the element file read in is the one used to create the grid file. The trait

```
6 0
trait opposite .TRT
Benz            No-Benz         Aromatic
CdO             No-CdO          Carbonyl
OsH             No-OH           Alcohol
N3              No-N3           Tert-Nitroger
NP              No-NP           Primary-Amine
COOH            No-CO2H         Carboxylic-Acid


6 9
trait opposite .GRD
Benz            No-Benz         Aromatic
 5 5 5 5 5 5 1 1 1
CdO             No-CdO          Carbonyl
 5 5 5 1 1 1 3 3 3
OsH             No-OH           Alcohol
 5 1 1 5 1 1 3 3 3
N3              No-N3           Tert-Nitrogen
 3 5 1 3 5 1 5 3 3
NP              No-NP           Primary-Amine
 3 3 3 3 3 3 5 1 1
COOH            No-CO2H         Carboxylic-Acid
 3 3 3 3 3 3 3 5 1
```

Text Frame 53: KEY Trait and Grid Files

name, its opposite, and description is the first data line, with each of the grid values for that trait are listed on the next line, separated by a space. These two lines of data are repeated for each trait.

V. Chiral Chromatography and CHIRULE

Chromatography relies on the differences in the internal energies of two compounds to effect separation. With enantiomers, however, virtually all of their physical properties (and hence internal energies) are identical. Chiral chromatography is based on having the enantiomers form diastereomeric complexes with either a chiral additive within the mobile phase or a chiral stationary phase. Once this complex is formed, the internal energies of the two enantiomers are no longer identical, and separation will occur.

Choosing the correct chiral column to effect such a separation is difficult. CHIRULE is a chiral column suggestion system. Its underlying principle proposes that if a new compound is chemically similar to an existing compound in the knowledge base, then the column and method for the existing compound would suggest a good starting point for developing a new method. Chemical similarity is determined by using molecular properties to model the chemical environment around the molecule's chiral carbon. A similarity search (using the molecular properties) yields a list of molecules called Sim-List molecules that are similar to the target molecule. A difference value, calculated from the property values of the target and Sim-List molecules, is

used to rank order the Sim-list molecules. The rank ordered list of molecules, along with the columns that have successfully separated the enantiomers, is then presented to the user as a good starting point in developing a method for separation of the target molecule.

This chapter describes the steps necessary for CHIRULE to suggest a molecule that is chemically similar to the researcher's target molecule. Sections include descriptions of the molecular properties used to model the chemical environment around the chiral carbon, the mechanisms behind the similarity searching procedure, calculations to determine the total difference measure used to rank order the results of the similarity searching, and a number of obvious engineering extensions to the work. Because of the detailed nature of several of these topics, several sections first present a general discussion of the methodology utilized, followed by a detailed explanation of how the procedure was implemented. Because of this, the reader will notice that some of the material in each section is presented twice in different ways. The first section provides an example of how CHIRULE can be used to suggest a chiral column to serve as a starting point in developing a method for separating an isomeric compound.

V.A. EXAMPLE - CHIRULE

Assume a new isomeric compound needs to be separated into its enantiomers. The literature would first be searched to see if a separation has already been reported for the target molecule. If no known separations are found, the traditional line of thinking would lead the researcher to one of several applications guides developed and distributed by chiral column suppliers. Based on functional groups contained within the molecule, the researcher is directed to a page within the application guide containing a number of molecules that have already been successfully resolved. The researcher makes a comparison between the target molecule and the molecules shown on that page. Once a similar molecule is chosen, the researcher is directed to another page that contains the method for that known molecule. This method serves as a starting point in developing a new method for the target molecule. This is the pattern upon which OPTIMER was built using the automated knowledge acquisition technique (see Chapter IV). The OPTIMER system has been expanded upon and improved, resulting in CHIRULE.

The basis for the classifications within the application guides is the functional groups that are present within the molecule. The concept employed is that if two different molecules (comparing the target molecule to a

molecule with a known separation in the application guide) have similar functional groups, then their enantiomers should be capable of separation using the same Chiral Stationary Phase (or CSP). It is the researcher's task to choose a molecule from the molecules grouped together on the page of the application guide which is similar to the target molecule.

CHIRULE attempts to automate this "similarity search" by modeling the chemical environment that surrounds the isomer's chiral carbon atom using several molecular properties (polarizability, hydrogen bonding, molecular volume). CHIRULE can suggest a column for use in developing a separation by applying the definition of similarity (based on molecular properties) to the traditional chiral column selection technique discussed at the beginning of this section. This modifies the traditional column selection approach as follows: a researcher has an isomeric compound to separate. Searching the literature does not yield a known separation for the researcher's target molecule. The researcher decides to consult CHIRULE, which contains a knowledge base of information about separations that have already been performed.

CHIRULE asks the user for the name of the MOL file to input. The target molecule is read in and fragmented into four molecular fragments around that chiral carbon by

replacing the bond to the chiral carbon atom with a bond to a hydrogen atom. Each molecular fragment is a molecule in and of itself, but is referred to as a molecular fragment to denote its origin and to prevent confusion with the original chiral molecule from whence it came. Various molecular properties used to model the chiral carbon's chemical environment are calculated for each of the molecular fragments. A similarity search is then performed to find other molecules, called Sim-List molecules, whose molecular fragments have property values that are similar to the target molecule. Once a list of the Sim-List molecules is obtained using this method, a difference measure based on the property values is calculated by comparing the target molecule to each Sim-List molecule. The difference measure is used to rank order the Sim-List molecules, which is then presented to the user as a starting point in developing a method for separation of the target molecule.

V.B. Molecular Properties

The key to the success of CHIRULE was choosing properties that would accurately model the chemical environment that the chiral carbon atom experiences. Inductive and steric effects are often mentioned when investigating the relationship between structure and reactivity. Polarizability, hydrogen bonding parameters,

and molecular volume were chosen as properties that quantified some aspect of these effects.

CHIRULE currently relies equally on these properties during the similarity searching process. Section V.E discusses cases where researchers may wish to weight certain properties differently. Note that the choice of these properties is not a limiting factor, but rather a reasonable starting point[114]. Section V.E also discusses the ease at which additional properties could be added to CHIRULE.

All properties were calculated based on molecular structure. SPARC was utilized to calculate values for the properties used in the similarity searching algorithm. Since the differences between two molecular fragment values are involved in the similarity search, most biases in SPARC's calculations are eliminated. Two issues regarding calculation of the molecular properties need to be considered: (1)what method is used to calculate the properties, and (2) do the calculated results seem reasonable. The methods used in the various calculations are summarized below. More complete descriptions are available in the SPARC Users Manual[68].

The computational approach for all properties used by CHIRULE is based on molecular-level descriptors, where the molecule-in-question is broken at each single bond. All

properties are expressed as linear combinations of the fragment contributions shown in Equation (7):

$$\chi^{\circ}(Molecule) = \sum_{j} (\chi_j^{\circ} - A_j) \qquad (7)$$

where χ_j° are intrinsic fragment contributions and A_j are adjustments relating to steric or electromagnetic perturbations from contiguous structural elements. In some instances, the χ° (molecule) are further adjusted for the specific property or solvent involved. Both χ_j° and A_j are based either on direct measurements of the property-in-question or on a directly related property for which large reliable data sets exist. Individual properties are calculated as follows.

V.B.1. Polarizability Volume

When a non-polar molecule is placed in an electric field, the resulting distortion of its nuclear position and electronic distribution will create a dipole moment within the molecule[115]. The magnitude of this induced dipole moment is proportional to the electric field (as long as the field is not too strong), as shown in Equation (8):

$$\mu_{induced} = \alpha E^{\bullet} \qquad (8)$$

where μ is the dipole moment, E^* is the electric field, and α is the polarizability of the molecule. When the field strength is expressed in V m^{-1}, the units of α are J^{-1} C^2 m^2. The awkwardness of these units leads most workers to convert polarizability to a polarizability volume according to the relation shown in Equation (9):

$$\alpha' = \frac{\alpha}{4\pi\epsilon_0} \tag{9}$$

where α is the polarizability, and α' is the polarizability volume. The units of $4\pi\epsilon_0$ are J^{-1} C^2 m^2, which leaves α' in units of volume. Values are usually quoted in either $Å^3$ or $10^{-24}cm^3$.

The expression for average molecular polarizability is shown in Equation (10):

$$\overline{\alpha}^{\circ} = \sum_i (\overline{\alpha}_i - A_i) \tag{10}$$

where α_i is the polarizability of fragment i and A_i are adjustments for the molecule-in-question, as explained below. Equation (11) further defines the average molecular polarizability for each fragment as:

$$\overline{\alpha}_i = \frac{1}{N_i}\left[\sum_j \chi_j\right]^2 \tag{11}$$

where χ_j are intrinsic atomic hybrid contributions, N_i is the number of electrons in fragment i, and the summation is over all atoms in fragment i. Hydrocarbon fragments with attached heteroatoms are adjusted by a 10% reduction in α_i. The χ_j are empirically determined from measured polarizabilities with the exception of hydrogen which is calculated from the measured polarizability of H_2.

In an effort to verify the molecular polarizability results obtained from SPARC, a number of Physical Chemistry texts were consulted (SPARC[116], Atkins[117], Castellan[118], Berry/Ross/Rice[119], Lowry/Richardson[120], Moelwyn-Hughes[121]) to obtain values for comparison. The two sections of Table VII contain the results of these efforts. The upper section contains comparisons with compounds that SPARC is able to calculate, whereas the lower section contains values useful in comparisons between the sources themselves. Table VII confirms that the values generated by SPARC are reasonable.

V.B.2. Hydrogen Bonding

SPARC uses the Taft-Kamlet approach to hydrogen bond modeling, resolving the interactions into proton donating (α) and proton accepting (ß) components which are presumed to be independently quantifiable. Hydrogen bonding requires the direct site coupling of a proton donating site of one

Table VII: Comparison of SPARC Polarizability Values

Molecule	SPARC	Atkins	Castellan	Berry	Lowry	Moelwyn
C_2H_5OH	5.0759		5.20		5.13	
C_6H_6	10.3565	0.104		10.40	10.39	9.890
CH_3OH	3.2454	3.23	3.00		3.26	
CH_4	2.6192	2.60	2.60			2.699
NH_3	2.0746	2.22	2.34	2.22		2.145
C_2H_6	4.4496		4.50			4.326
C_3H_8	6.2801		6.40			
$HCONH_2$	4.0660				4.22	
HCOOH	3.4022				3.39	
Ar		1.66	1.63			1.626
CO		1.98		1.98		1.926
CO_2		2.63		2.63		2.594
H_2		81.90	0.81	0.819		0.802
H_2O		1.48	1.59		1.48	1.444
N_2		1.77	1.72	1.77		1.734
HBr		3.61				3.492

molecule with a proton accepting site of another molecule. Solute parameters are calculated as pseudo pKa's with the electrostatic component treated as a dipole transition. Any given solute can be represented by Equation (12):

$$\alpha = \sum_i \alpha_i \qquad \beta = \sum_i \beta_i \tag{12}$$

where α_i represents potential proton donor sites and $ß_i$ represents potential proton acceptor sites within the molecule. Chemical process modeling uses the term "reaction centers" (denoted C) to describe potential sites for a reaction of interest, and the term "perturbed" structure (denoted P) to represent a molecular structure appended to C. The potential reaction centers for hydrogen bonding corresponds closely with Brönsted acid (α) and base (ß) sites. Using these definitions, Equation (13) can be used to describe a given reaction center:

$$\alpha = \alpha_c + \delta_p^{\alpha} \qquad \beta = \beta_c + \delta_p^{\beta} \tag{13}$$

where α_c and $ß_c$ denote intrinsic behavior of the reaction center and δ_p^{α} and $\delta_p^{ß}$ denote changes in reactivity effected by the perturber structure. The electrostatic and resonance susceptibilities along with the reaction center parameters α_c and $ß_c$ are based on Kamlet-Taft hydrogen bonding constants[122].

V.B.3. Molecular Volume

Equation (14) shows the expression for the density based molecular volume:

$$V_{25}^{\circ} = \sum_i (V_i^{frag} - A_i) \qquad (14)$$

where V_i^{frag} is the volume of fragment i and A is a correction to that volume based on the number and size of fragments attached to it. Adjustments for shrinkage due to dipolar interactions and expansion due to ordering by hydrogen bonding are made to the zero order volume at 25° C according to Equation (15):

$$V_{25} = V_{25}^{\circ} + A_d \left[\frac{\sum_i Dipole_i^2}{V_{25}^{\circ}} \right] + A_H \sum_i \alpha_i \sum_i \beta_i \qquad (15)$$

where A_d is the correction due to dipolar interactions and A_H is the correction due to ordering by hydrogen bonding in both proton donating (α) and proton accepting (ß) fragments. The volume at any temperature, T, is expressed as a polynomial expansion in (T-25) shown in Equation (16):

$$V_T = V_{25} \left(1 + \sum_n a_n (T-25)^n \right) \qquad (16)$$

where the definitions are as above.

Table VIII: Comparison of SPARC Molecular Volume Values

Molecule	Temperature	SPARC	Calculated
CH_4	-164	36.15	34.42
	20	63.25	88.13
CH_3CH_3	-108	49.22	52.57
	-60	55.66	59.08
	25	66.97	-
$CH_3CH_2CH_3$	-45	72.47	75.36
	25	83.38	-
$CH_3CH_2CH_2CH_3$	0	95.10	96.67
	20	98.69	100.41
	25	99.59	-
CH_3OH	20	40.16	40.48
	25	40.16	-
CHOOH	20	42.37	37.73
	25	42.69	-
$CHONH_2$	20	35.52	39.74
	25	35.62	-
$CH_3CH_2NH_2$	20	66.62	66.03
	25	67.09	-
C_6H_6	20	88.27	80.91
	25	88.87	-
C_6H_5-C_2H_5	20	122.31	122.46
	25	123.17	-
C_6H_5-$CONH_2$	4	92.18	90.35
	25	92.18	-

The molecular volume values calculated by SPARC for a number of molecular fragments were compared with values obtained by dividing the molecular weight by the density. Molecular weight and density values were obtained from the literature[123]. The molecular fragments used were chosen from those in the Fragment Value KB (see Appendix G). Remember that molecular fragments are in fact molecules in and of themselves, but are arrived at by fragmenting an enantiomer around its chiral center. Table VIII shows these comparisons, and confirms that the results generated by SPARC are reasonable.

The beta release of SPARC had difficulties calculating the molecular volume of Imidazoles. Molecular volume values in these cases were arrived at by dividing the molecular weight by the density of the molecular fragment as described above. This is to be corrected in SPARC's final release.

Values for each property are calculated for the molecular fragments that make up the target molecule. A similarity search is then performed using these results to create a possible hit list of compounds that have been found to be chemically similar. The choice of polarizability, hydrogen bonding parameters, and molecular volume is not necessarily a inherent property of CHIRULE. Other properties could be added, or substituted for the ones chosen (see Chapter V.E below).

V.C. Similarity Searching

A similarity search is used to generate a list of molecules from CHIRULE's Molecule KB that are similar to the researcher's target molecule. This process can be broken down into four stages (1) adding the molecule to all KBs, (2) generating the hashed data set, (3) use the hashed data set to create the list of similar fragments, and (4) generate the list of similar molecules. Figure 10 gives an overview of the similarity searching procedure by showing a detailed list of the tasks performed in every stage.

Before delving into the individual tasks, some definitions are necessary. Each "Sim-List" molecule, a molecule chosen from the Molecule KB, is compared with the target molecule on a fragment by fragment basis looking for similarities in several molecular properties. Rather than searching fragment by fragment through the entire knowledge base, a hashing function is utilized to greatly enhance the speed of the search. The total difference value (see Table IX) is the sum of the difference values calculated for each fragment pair (one Sim-List and one target fragment) for all of the molecular properties used by CHIRULE (polarizability, proton donating and withdrawing contributions to hydrogen bonding, and molecular volume). The measure is used to rank order the list of similar molecules. Since the similarity search process requires

(1) Add molecules to all KBs by:
- Reading in the MOL file, adding it to the Molecule KB
- Fragment the molecule, add it to the Molecular Fragment KB
- Calculate the 4 molecular properties for each fragment, add them to the Fragment Value KB

(2) Generate the hashed data set by:
- Reading in each value from the Fragment Value KB
- Divide it by the property adjustment value
- Multiply by the Delta Value
- Round the result up and down
- Make two entries in the Fragment Value Range KB

----> **Choose a Target molecule**

(3) Create the list of similar fragments by:
- Comparing the fragments of the Target Molecule to each Sim-List molecule
- Find and remove fragments that appear in both the target molecule and the Sim-List molecule
- Compare remaining fragments using hashing bins
- Make an entry in the Fragment Match KB if the target molecule fragment resides in the same bin as the Sim-List molecule fragment

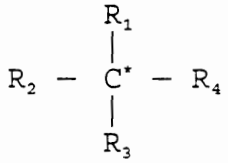
(4) Generate the list of similar molecules by doing the following for each similar fragment pair:
- Get the property value difference
- Multiply by the offset
- Divide by the normalization value
- Add the four property difference values to get the Total Difference Value for the fragment pair

Then continue with each of the fragment pairs:
- Assemble the correct number of fragments
- Sum all their Total Difference Values to arrive at the Total Difference Measure
- Assert the Total Difference Measure to the Column Suggestion KB
- Convert the Column Suggestion KB to a list
- Order this list using the Total Difference Measure, with the lowest value first
- Print out the list of column suggestions

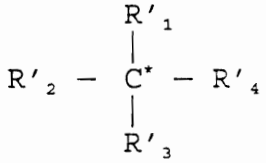
Figure 10: Similarity Searching Procedure

Table IX: Similarity Searching Definitions

Target Molecule



Sim-List Molecule



Molecular Fragments:

R_1	R'_1
R_2	R'_2
R_3	R'_3
R_4	R'_4

Difference Value for One Fragment Pair X_1 (R_1 & R'_1):

$$\left| \text{Pol. } (R_1) - \text{Pol. } (R'_1) \right|_{ABS} = \text{Diff. Value}_{\text{Pol.}} (X_1)$$

Total Difference Value for Fragment pair X_1:

$$\begin{aligned}
&\left| \text{Pol. } (R_1) - \text{Pol. } (R'_1) \right|_{ABS} &&= \text{Diff. Value}_{\text{Pol.}} (X_1) \\
&\left| \text{Alpha } (R_1) - \text{Alpha } (R'_1) \right|_{ABS} &&= \text{Diff. Value}_{\text{Alpha}} (X_1) \\
&\left| \text{Beta } (R_1) - \text{Beta } (R'_1) \right|_{ABS} &&= \text{Diff. Value}_{\text{Beta}} (X_1) \\
&\left| \text{Vol. } (R_1) - \text{Vol. } (R'_1) \right|_{ABS} &&= \text{Diff. Value}_{\text{Vol.}} (X_1) \\
& && \overline{\qquad\qquad\qquad\qquad} \\
& && \text{Total Difference Value } (X_1)
\end{aligned}$$

Total Difference Measure Between the Two Molecules:

Total Difference Value (X_1)
Total Difference Value (X_2)
Total Difference Value (X_3)
Total Difference Value (X_4)

Total Difference Measure

that the target molecule be added to all knowledge bases before the search proceeds, an exact match with itself is explicitly ruled out (see Section V.C.4.a below). Note that Table IX shows only one possible permutation in comparing fragment (R_x) with the fragments of the Sim-List molecule (R'_x). The sections below discuss these topics in greater detail.

V.C.1. Hashing Data Sets

In searching for a match between an unknown and members of an unhashed data set, a time consuming item by item search is required. Hashing a data set involves dividing a set of data into a discrete number of bins based on each item's value for a certain property[124]. Once the data set is hashed, only items in a single bin (or adjacent bins) need to be considered, greatly reducing search times. Each item's property value is represented as a point on a line that encompasses all of the possible values of that property (see Figure 11). The hashing function divides the line into a number of segments known as bins. Any item whose property value falls on the line segment contained within a certain bin is considered to be a member of that bin.

The hashing function controls both the number and the size of the bins. The simplest hashing function is one that divides the possible range of values into a series of

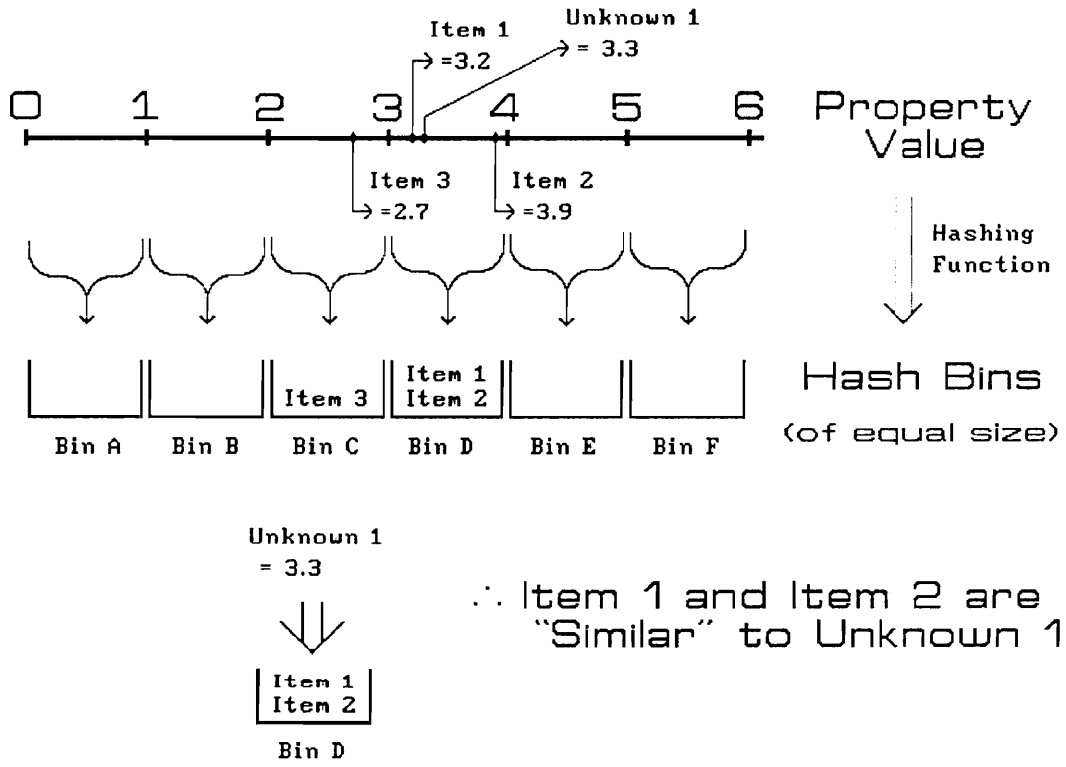


Figure 11: Hashing Functions

identically sized bins (see Figure 11). Since the length of the line is defined by the largest property value among the items, defining the number of bins into which to divide the line also defines the width of the bins. As the number of bins increases for any given property, the length of the line segment that is included in each bin decreases, and the bins width decreases. In cases where the data is clustered around one or even several values, hashing functions which generate bins of unequal size must be employed.

While the exact number of bins is not critical, a good rule of thumb is that virtually all bins should contain between two and ten items. Generally, as the width of the bins decrease, the number of items found within the bin should also decrease. The design of a hashing function must avoid the two extreme cases. Having too many items in each bin results in a loss of searching efficiency, while having only one or two items in each bin may not provide enough data to be reviewed.

Once all members in the data set have been hashed into different bins, the search process is reduced to two steps: 1) apply the hashing function to the unknown item to determine the appropriate bin, and 2) select all of the items in that bin for use in a final search. Referring to Figure 11, assume the unknown has a property value of 3.3. For this example, only 3 items in the data set are shown.

Hashing the unknown indicates that it belongs to bin D, which yields items 1 and 2 as matches.

This single example does not adequately represent the time savings that hashing a data set can produce. The amount of time required to hash a data set is approximately identical to the amount of time it takes to do a single item by item search, since both cases result in every item in the data set being accessed. The multiple searches required for considering the 16 possible combinations of the 4 fragments of the target molecule with the 4 fragments of each Sim-List molecule makes hashing the data set beneficial.

V.C.2. CHIRULE's Hashing Function

The similarity search employed involves finding molecules in CHIRULE's Molecule KB whose fragments have property values similar to the fragments of the target molecule. While a direct comparison of molecules would involve a lengthy fragment by fragment search, hashing the Fragment Value KB into bins greatly increases the similarity searches efficiency and speed. Every molecule in CHIRULE's Molecule KB is made up of four molecular fragments. There is one entry in the Fragment Value KB representing each individual molecular fragment for every molecular property. Thus, a direct comparison of the target molecule with any one particular molecule in the Molecule KB could result in

as many as 16 individual property by property searches (assuming the two molecules share no identical fragments, 4 fragments/molecule x 4 properties/fragment = 16). Hashing is critical to the success of the similarity search based on molecular properties.

The Fragment Value KB contains 85 molecular fragments that, when combined, make up the 96 molecules in the Molecule KB. Note that each fragment will appear 4 times in the Fragment Value KB (one entry for each of the molecular properties used), which has a total of 340 entries (4*85). Hashing the value of each entry in the Fragment Value KB generates the Fragment Value Range KB (for a description of all knowledge bases, see Chapter VII.B and Appendix G). The first hashing function utilized took advantage of the fact that all property values in the Fragment Value KB ranged from 0 to 2.5. Simply shifting the decimal point one place to the right, truncating, and using the resulting integer values as bin numbers distributes the range of values for a single property into 25 bins. This way the 85 unique molecular fragments can be hashed into 25 bins without violating either of the extreme cases mentioned above. This uncomplicated hashing function yielded such satisfactory results that it has only been slightly modified for use in the CHIRULE system discussed below.

A closer look at Figure 11 demonstrates this modification. Hashing an unknown with a property value of 3.3 indicates that the item belongs to bin D, which contains item 1 and item 2. Item 1 (value 3.2) found in bin D is a very close match to the unknown (value 3.3). However, item 3 (value 2.7) also found in bin D is numerically as close to the unknown as item 2 (value 3.9) found in bin C which would not be chosen. This is a result of the boundaries imposed by creating the bins. Regardless of the width of the bin, there exists the possibility that an item that is on the other side of a bin boundary is as close or closer than an item that is contained within the bin. The hashing function used overcomes this by assigning each entry to two adjacent bins.

The hashing function adjusts all property values in the Fragment Value KB before being hashed so that their values fall between 0 and 10 (see Text Frame 54). This is done so that the same hashing function can be used for all of the properties. If the original hashing function were applied to molecular volume, 1,800 bins would be generated (the highest molecular volume, 180 times 10). It is unlikely that many of the fragments would reside in the

Property	Range of Values	Adjustment Value
Polarizability	0-2.5	1
Hydrogen Bonding		
Alpha	0-1.57	1
Beta	0-1.77	1
Molecular Volume	18-180	100

Text Frame 54: Property Adjustment Values

same bin since there are only 85 molecular fragments. This adjustment allows the set of fragment values for any one property to be hashed into approximately the same number of bins. For molecular volume, the values in the Fragment Value KB are divided by 100 to bring them into the same range as the values for the other molecular properties. The other property values do not need to be adjusted, since their values already fall between 0 and 10 (see Text Frame 54). Once all properties fall within the same range, the program has been written such that the delta value can be used to modify the number of bins created for all properties.

The final hashing function utilized (see Figure 10) takes each member of the Fragment Value KB and multiplies its adjusted property value by the current delta value. The delta value defines both the size and number of bins that the hashing function generates and is derived from the original function (a delta value of 10 would be equivalent to shifting the decimal place). Since all property values are adjusted so they fall between 0 and 10, a delta value of 1 would result in no more than 10 bins (because the hashed values would fall between 0 and 10). A delta value of 10, currently employed by CHIRULE, would result in no more than 100 bins.

The product of the delta value and the adjusted property value is rounded to the next smaller and next larger integer number. The numbers that result become the bin designations, and are used to assign the fragment to two adjacent bins to overcome the bin boundary problem explained above. An entry is created for each number in the Fragment Value Range KB by the predicate "gen_fvrange_db". The hashing function is used to convert all of the entries in the Fragment Value KB into entries in the Fragment Value Range KB (the hashed data set).

The delta value of 10 used in this work was derived from the initial hashing function that simply shifted the decimal one place to the right. Results showed that this simple hashing function was sufficient to adequately distribute the 85 different fragments into the 25 bins. While researchers can experiment with other delta values (see discussion in Chapter VIII), the sheer simplicity of the delta value justified its use to show feasibility in this work.

V.C.3. CHIRULE's Similarity Search

There are two areas to address concerning the similarity searching process utilized by CHIRULE. The first section describes how CHIRULE utilizes the similarity searching process to come with a list of similar molecules.

This is followed by a detailed explanation of the similarity search is then presented, describing each Prolog predicate involved in the search process. Note that it is necessary to cover some information in both sections.

V.C.3.a. Overview

There are two tasks that must be completed before CHIRULE can perform a similarity search based on the molecular properties. First, the target molecule must be added to several of CHIRULE's KBs and then the Fragment Value KB must be hashed to generate the Fragment Value Range KB. The Prolog predicate "chirule" used to run the CHIRULE system performs both tasks before attempting the similarity search.

In preparation for the search, the target molecule's MOL file must be read in and added to the Molecule KB, with its fragments added to the Molecular Fragment KB. Each of the four molecular fragments must have all of their molecular properties calculated, which are then added to the Fragment Value KB. This is done so that when the Fragment Value Range KB (the hashed data set) is generated from the Fragment Value KB, the target molecule will be included.

Once the Fragment Value Range KB is generated, a molecule is chosen from the Molecule KB. This molecule, referred to as a "Sim-List" molecule, will be used by the

similarity search for comparison to the target molecule. Because of hashing, the similarity search is reduced to taking a fragment from both the target molecule and the Sim-List molecule and determining if both reside in the same bin within the Fragment Value Range KB. If the fragments are found to reside in the same bin for at least one of the properties, an entry is made in the Fragment Match KB and another set of fragments is chosen for comparison. If the fragments do not reside in the same bin, the target molecule's fragment is compared with a different fragment from the Sim-List molecule.

The process of comparing the fragments of the target molecule to the Sim-List molecule becomes a mathematical combination problem. Prolog's relentless searching and automatic backtracking handles the 24 possible combinations between the 2 sets of 4 fragments (4 factorial = 24) automatically. The resulting Fragment Match KB contains all of the Sim-List molecules which were found to contain similar fragments with the target molecule. This information must be condensed into a rank ordered list of similar molecules.

V.C.3.b. Implementation

Prolog refers to subroutines as predicates (see Chapter VI). All predicate names are double quoted and in lower

case. Consult module "CHIRULE.HC" in Appendix F for the actual code contained in each of Prolog predicates.

Finding similar molecules is accomplished using the predicate "msimilar" (molecule-similar). This predicate takes the molecular fragments of the target molecule and compares it to the Sim-Fragments (the molecular fragments of a Sim-List molecule). After verifying that the two molecules are not the exact same molecule, both sets of molecular fragments are converted into separate lists, each consisting of four members. Any molecular fragments that are found to be an exact match in the two lists are removed by the predicate "removedups". What remains are two lists containing fragments which need to be compared to see if they are similar.

These two lists are passed by "msimilar" to predicate "fsimilar" (fragment-similar) to see if fragments can be found in both lists which are similar. "fsimilar" takes one fragment at a time from the target molecule's fragment list and calls predicate "psimilar" (property similar) in an attempt to find a similar fragment in the Sim-Fragment list. Predicate "fsimilar" will fail if it cannot find a matching Sim-Fragment for all molecular fragments found in the target molecule. If "fsimilar" fails, the current Sim-List molecule will not be added to the Fragment Match KB, another

Sim-List molecule will be selected, and the similarity search will proceed.

Predicate "psimilar" compares the fragment from the target molecule to the first Sim-Fragment. The two fragments passed by "fsimilar" are compared by examining the hashed property values (the Fragment Value Range KB). If both fragments are members of the same bin **for any property**, they are viewed as being similar, and an entry is added to the Fragment Match KB. This similar fragment is then removed from the Sim-Fragment list. Predicate "fsimilar" then takes the next non-identical fragment from the target molecule's fragment list and again calls predicate "psimilar" with the remaining Sim-Fragments. This process is repeated until all fragments in the target molecule's fragment list are matched with Sim-Fragments. If a similar fragment cannot be found, the current Sim-List molecule is not similar to the target molecule and the process is attempted again with another Sim-List molecule chosen from the Molecule KB.

Notice that predicate "msimilar" has a 'fail' statement as in its last line. To understand why the fail statement is necessary, a review of the searching predicates is necessary. When predicate "msimilar" successfully executes and reaches the fail statement, predicate "fsimilar" has already taken each fragment in the target molecule and

either 1) found an identical fragment in the Sim-List molecule or 2) called predicate "psimilar" to find a similar fragment in the Sim-List molecule. Predicate "fsimilar" in effect pairs off the non-identical fragments. Predicate "psimilar" then takes these pairs of fragments and looks in the Fragment Value Range KB (the hashed data set) to see if each resides in the same hash bin for **ANY** property (see above). Once an entry in the Fragment Match KB is made for the fragment pair's first similar property, there may be other properties that are also similar. The 'fail' statement in predicate "msimilar" insures that all of these similarities are found and added to the Fragment Match KB (see Chapter VI for a further discussion of relentless searching).

At this point, each entry in the Fragment Match KB contains the following information: the two fragments that were found to be similar, the molecules from which each came (the target molecule and the Sim-List molecule), the number of fragments that had to be matched, the property for which the match was found, the delta value used to generate the Fragment Value Range KB (the hashed data set), and the difference value for the two fragments for that property. The difference value is arrived at by taking the absolute value of the difference between the two fragments values for the property, multiplying that difference by an offset

value, and dividing by the delta value discussed above. Because this difference is typically quite small, an offset value is used to shift the decimal place (see predicate diff_offset). The delta value is included in the difference value because it represents the width of the bins created by the hashing function. As the bin size decreases, the range of values contained in the bin will be smaller, and each fragment in the bin will be a closer match. The delta value is included in the Fragment Match KB in addition to the difference value to permit the comparison of results generated using different delta values (or bin sizes).

V.C.4. Generating Column Suggestions

The final task in the similarity search is to combine all the fragments for all Sim-List molecules represented in the Fragment Match KB and calculate the total difference measure. The total difference measure is placed, along with the target molecule and the similar Sim-List molecule, into the Column Suggestion KB. Next, the Column Suggestion KB is converted into a rank ordered list of column suggestions according to the total difference measure and presented to the user.

There are three factors that need to be accounted for in calculating the total difference measure: the first involves combining the fragments in the Fragment Match KB

into matching molecules, the second requires the normalization of each property, and the third covers the different permutations possible when comparing the fragments. The calculation of the difference value for two fragments with respect to one property is contained in the Fragment Match KB, and was described in the previous section. This must be expanded to include not only all of the properties for that one pair of similar fragments, but also all of the other fragment pairs found when comparing the target molecule with the Sim-List molecules.

V.C.4.a. Identical Fragmentsn

The similarity search involves comparing the four molecular fragments in the target molecule to a Sim-List molecule. It is likely that one or more of fragments will be identical, meaning the same fragment will exist in both molecules. The number of identical fragments must be accounted for in the final difference value, since a Sim-List molecule with three identical fragments in common with the target molecule is likely to be a much better match than a Sim-List molecule that has only one identical fragment.

When searching for a Sim-List molecule similar to the target molecule, one of four conditions exist: 1) three of the fragments will be identical leaving one fragment of each molecule to test for similarity; 2) two identical fragments

with two fragment pairs to test; 3) one identical fragment with three fragment pairs to test; and 4) no identical fragments with four fragment pairs to test. The other case that would be possible, that of four identical fragments, is ruled out in the similarity search because it indicates the target molecule is identical to a Sim-List molecule already found in the KB.

The mechanism for calculating the total difference measure must handle the total difference values from each matching fragment pair. This is accomplished by adding together the difference values calculated for each similar fragment pair (see Table IX). As a result, the first condition described above (where there are three exactly matching fragments and only one similar fragment) should yield the lowest total difference measure. Note that the difference value for identical fragments is zero, and hence does not need to be calculated.

V.C.4.b. Property Normalization

The total difference measure is based on subtracting individual fragments property values for a number of molecular properties. Each of the four properties used in this work should have the same contribution to the total difference measure (for a different approach, see Section V.D.3). However, their individual range of values vary (see

Text Frame 55). As a result, the difference values for each of the properties were normalized based on the individual property's range of values. For example, the molecular volume values for the molecular fragments in CHIRULE range from 18 to 200, while the polarizability values range from 0.12 to 2.46 (See Appendix G). The difference values generated for these two properties will differ by two orders of magnitude. If they were not normalized, the molecular volume contribution would overwhelm the contribution of polarizability in calculating the total difference measure

Molecular Property	Range of Values	Normalization Value
Polarizability	0-2.5	3
Hydrogen Bonding		
Alpha	0-1.57	2
Beta	0-1.77	2
Molecular Volume	18-180	200

Text Frame 55: Property Normalization Values

The normalizing factors are derived from each of the properties knowledge bases. Each normalizing factor is used to adjust all fragment's property values so they range between 0 and 1, as shown in Text Frame 55. For example, all molecular fragments have polarizabilities between 0 and 3. The normalizing factor used for the polarizability property is 3 (see predicate "p_norm" in module CHIRULE.HC in Appendix F).

The normalization factors are utilized by predicate "column_value". Predicate "calc_diff" calls "column_value" with each similar fragment pair. "column_value" calculates the difference value for each property, normalizes them, and

adds them together, returning the total difference value for the similar fragment pair to "calc_diff" (see Table IX).

V.C.4.c. Permutations

The final factor that needs to be addressed involves comparing the target molecule with a Sim-List molecule where both molecules have two, three, or four similar fragments. Choosing molecular fragments, one from each molecule, can be done in a number of ways and can be characterized as a mathematical permutation problem. While there are only two possibilities when matching two fragment pairs, that number increases to six with three fragment pairs and to 24 for four fragment pairs (a factorial relationship). Because of the possible permutations, there will most likely be cases where two molecules will be found to match using different arrangements of the functional groups. When this occurs, only the arrangement with the lowest total difference measure is saved in the Column Suggestion KB.

These three factors (the number of non-identical fragments, normalizing the properties, and the permutations involved in comparing the fragments) are combined by the predicate "calc_diff" (see module "CHIRULE.HC" in Appendix F). Each instance of the predicate accommodates a different number of non-identical fragments (the first factor

described above) occurring between the target molecule and the Sim-List molecule.

The first task in each instance of "calc_diff" is to assemble the requisite number of fragment pairs, so the fragment's difference values can be calculated. This is done by querying the Fragment Match KB for a Sim-List molecule that was found to be similar to the target molecule. Using the instance where there are 2 non-identical fragment pairs, the Fragment Match KB is again queried for another fragment pair for the same molecules. After the second pair is tested to make sure it is different from the first pair, predicate "calc_diff" determines the difference value for each of the matching fragment pairs by calling predicate "column_value". If a second fragment pair is not found in the Fragment Match KB, that particular Sim-List molecule is not considered as being similar to the target molecule.

Predicate "column_value" (see module "CHIRULE.HC" in Appendix F) individually calculates the difference value between two matching fragments for each of the four properties utilized by CHIRULE. Each of these individual property difference values is calculated by the predicate "get_property" (see module "CHIRULE.HC" in Appendix F). If there is an entry in the Fragment Match KB for that property and the two fragments (i.e. the similarity search determined

they were similar), the value is obtained there. If there is no entry, the actual difference is calculated by "get_property" from entries in the Fragment Value KB. The predicate "column_value" then normalizes each of the properties (the second factor mentioned above) and combines them to yield the total difference value between the two matching fragments.

The other task performed by predicate "column_value" is to limit the searching of the fragment KB by placing a limit on the size of the total difference measure. In an effort to speed up the similarity search process and to report results that would be useful to the user of CHIRULE, an arbitrary limit can be placed on the values that are reported. Results have shown that limits between 1000 and 2000 are reasonable (see Chapter VIII.A). The total difference measure is summed first across the molecular properties and then across each of the fragments. As soon as the limit is reached, that particular permutation is abandoned.

After the difference values for all of the properties have been calculated and combined into the total difference measure, an entry into the Column Suggestion KB for the target molecule and the Sim-List molecule is made by the predicate "assert_colsugg" (see module "CHIRULE.HC" in Appendix F). If an entry in the Column Suggestion KB has

not been made previously for the two similar molecules, a new entry is created. If one already exists, that means that a previous permutation of similar fragments was found. This entry is replaced only if the new difference value is less than the previously determined value, thus saving only the permutation of the fragments with the lowest total difference measure (the third factor mentioned above).

Once the Column Suggestion KB is created, all of the information contained in it needs to be sorted in ascending order according to the match value and presented to the user. The predicate "make_col_list" converts the Column Suggestion KB into a list of the suggestions, which is then sorted by the predicate "order_col_list". This ordered list is then presented to the user by the predicate "print_col_list".

It should be noted that CHIRULE is not limited to proposing any particular type of chiral column. Chapter II.A.3 uses the differences in separation mechanisms to describe the various types of chiral stationary phases that are currently available. No chiral column types are excluded because the similarity search is looking only at the chemical environment around the chiral carbon, not at the interaction that occurs between the chiral molecule and the CSP. It is possible that for any one given property, certain types of columns would not be predicted as

accurately. For example, polarizability would not describe the steric factors that play an important role for the inclusion of chiral molecules in cavity columns. Using molecular volume, in addition to polarizability, better represents the chemical environment of chiral molecules that are resolved using cavity type columns. The predicting capabilities of CHIRULE has been greatly expanded by utilizing a number of properties and combining the results.

V.D. Extending CHIRULE

There are a number of issues raised by this work that are outside the bounds of the dissertation research. Several of these issues have been addressed here rather than in latter chapters because they are viewed to be obvious extensions of the work rather than projections into future research.

V.D.1. Adding Properties

The first of these issues concerns adding other properties. Four molecular properties were chosen in an effort to model the inductive and steric effects involved in structure reactivity relationships. During the initial development stage, the properties chosen only quantified the inductive effect (polarizability and hydrogen bonding parameters). In an effort to expand the capabilities of

CHIRULE to take into consideration steric effects and to demonstrate the ease with which properties could be added, molecular volume was added. The modifications required in adding an additional property, summarized in Table IX, can be divided into three categories: changes to the CHIRULE system, changes to the molecular property acquisition routines for SPARC, and additions to the knowledge base routines.

V.D.1.a. Changes to CHIRULE

The changes to the CHIRULE system that are required to add an additional property are made in the module CHIRULE.HC (see Appendix F). The first change is to the predicate "column_value", where an additional call to predicate "get_property" needs to be added. The additional call will yield the difference value between the two fragments specified for the new property just added. Another new predicate "x_norm ?x" must be added where ?x is the normalization factor for the new property "x". The difference value returned by "get_property" must be normalized by dividing the measure by the normalizing factor ?x.

The steps outlined above include the new property in the similarity search, adding the difference value for the new property to the total difference measure. The final

Table X: Steps for Adding Properties to CHIRULE

Prolog Module "CHIRULE.HC"

"show_chiral ?name" - Displays all the information contained within the various knowledge bases for the molecule ?name. Alterations include adding queries to obtain the new properties value for each fragment, and a print statement to display them.

"column_value ?ssa ?ssb ?nmatch ?frag ?frag2 ?oldtot ?value ?newtot" - Obtain all property values for fragment pair ?frag/?frag2, calculate the difference value ?value, add it to the old total and pass back the new difference measure. Alterations include adding another call to predicate "get_property" for the new property, normalizing the returned value, and adding it to the values from the other properties to obtain ?value.

"x_norm x" - New normalization factor for new property x.

Prolog Module "SPARC.HC"

"getall_sparc ?name" - Gets all property values for the four fragments in molecule ?name. Calls the following three predicates.

"checkfrag ?name ?ss" - Checks to see if the four fragments for molecule ?name (with SMILES String ?ss) already have values for all properties in the fragment value KB. Additions must be made to check all four fragments for the new property.

"sparcfrag_batch ?ss" - Writes the required information to the SPARC batch mode input file BATCH.DAT to calculate all properties for fragment SMILES String ?ss. An additional line must be added to provide the default information for the new property.

"getsparc_info ?pol ?alpha ?beta ?vol ?x" - Reads all records from the SPARC batch mode output file BATCH.OUT for one fragment, returning all properties for that fragment. Additions include an additional parameter for the new property, and an additional call to predicate "getsparc_record".

Prolog Module "RES.HC"

"load_value_k x" - Load all of the molecular fragment KBs. Add a line, supplying the file name for the new property.

"gen_all_range ?delta" - Generates all of the bins for the similarity searching. Add a line to generate the bins for the new property from the fragment value KB.

change required in module CHIRULE.HC is to the predicate "show_chiral" which prints out the 4 molecular fragments and their property values. Calls to the Fragment Value KB "fg-valu" for each of the fragments must be added, as well as printing out each fragments value for the new property. This change, like the others above, are all analogous to the existing Prolog code.

V.D.1.b. Interactions with SPARC

The second modification involves the changes necessary to obtain property values for fragments from SPARC, the molecular property calculator used in this work. The predicate "getall_sparc" in module SPARC.HC (see Appendix F) contains calls to the three predicates that need to be modified to accept a new property. The first predicate, "checkfrag" determines whether values for all of the properties already exist for each of the fragments. The additional calls to the Fragment Value KB for the new property can be added using the existing code as an example. The second predicate "sparcfrag_batch", writes the information necessary to calculate all of the properties for one fragment to the batch mode input file BATCH.DAT. Changes required here involve adding the information for the new property, and can be made using the code for existing properties as an example.

If an application developed in CHESS necessitates acquiring properties for individual molecules (as apposed to fragments), changes would have to be made to the predicate "sparc_batch" to write the appropriate information into the batch mode input file "BATCH.DAT". The existing work can be used as a template for the required changes. The predicate that reads the information generated by SPARC, "getinfo", is not effected by adding new properties. "getinfo" will read in the new property value without necessitating any changes.

V.D.1.c. Knowledge Bases

The final changes that are required pertain to the routines that deal with the knowledge bases that have already been developed and can be found in module RES.HC (see Appendix F). Predicate "load_value_kb" requires an additional line (similar to the existing lines) to load the values for the new property value KB. Also, a line similar to those existing should be added to the predicate "gen_all_range" to generate the bins to be used in the similarity search.

After making these changes to the software, the procedure for adding another property to the CHIRULE system is as follows. After deciding on the property to be added, execute the predicate "update_fg-value" from within the CHESS system. This will generate the SPARC batch mode input

file BATCH.DAT and read back in the results file BATCH.OUT after SPARC has generated it. At this point, exit Prolog, make sure the changes indicated above have been made, and start up CHIRULE again. When the knowledge bases are loaded, the new property will also be loaded and available to CHIRULE.

Notice that in the procedure described above, there is no strict requirement that the property values for the fragments come only from SPARC. The property values could be created by other property estimators, provided that the Fragment Value KB is generated in the same format. This allows for the flexibility to add other pertinent properties as they are discovered, such as Hausdorff's chirality measure as described in Chapter II.A.5.

V.D.2. Chiral Separation Databases

The Molecule KB and the Chiral Column Suggestion KB do not have to be expanded manually as they have been in this work. It is possible to take advantage of currently existing databases of chiral separations, like ChirBase described in Chapter II.A.1. This task would be made easier in this case because ChirBase uses ChemBase, which is a sibling product to ChemTalk used in this work. Queries could be easily written to include both of the above knowledge bases in the CHIRULE system.

V.D.3. Property Normalization Factors

A final extension to this work concerns the use of normalization factors to permit the combination of the various properties utilized (see section V.C.4.b above). In order to prove feasibility of the concepts presented, this work treats each of the properties as contributing equally in the similarity searching process. The normalization factors could be utilized in such a way as to permit one or several of the properties to be emphasized (or de-emphasized) to study the contribution of each property to the total difference measure. Thus if a researcher had reason to believe that steric factors were the most important contributor for the target molecule, the properties modelling steric interactions could be treated preferentially. This would only require changing the normalization factor for that property so that it makes a greater contribution to the total difference measure.

VI. Prolog

There is a wide spectrum of tools available for building expert systems. Two common groupings used to categorize these tools are programming languages and expert system shells. Expert system shells provide the programmer with a variety of utilities such as predefined knowledge representations, control structures, debugging aids, and knowledge base construction and maintenance techniques. There are a wide variety of shells commercially available, each having their own strengths and weaknesses. Languages such as Prolog and LISP provide the expert system programmer with the flexibility to develop a knowledge structure that accurately represents the problem area rather than having to mold the problem into the structures dictated by an expert system shell. It is not the purpose of this Chapter to provide a review of the tools used to construct Expert Systems. The reader is directed to the literature and to Chapter II.B.6-7 for further information.

Instead, this chapter will examine a number of key topics underlying the reason Prolog was chosen to implement CHIRULE. Determining the Index of Hydrogen Deficiency[125] will serve as an example to demonstrate these topics. The number calculated represents the amount of unsaturation found within the molecule, which translates into the number

of rings, double, and triple bonds the molecule contains. The Index of Hydrogen Deficiency is derived from the general molecular formula for unsaturated hydrocarbons (C_nH_{2n+2}) and can be converted for use with all molecules, as shown in Equation (17):

$$\frac{(2 * Tetravalent + 2 + Trivalent) - Monovalent}{2} = Index \quad \textbf{(17)}$$

Text Frame 56 contains several examples demonstrating Equation (17). A later section will demonstrate how this formula can be applied to a molecule's MOL file to determine the number of rings within that molecule.

MOLECULE	FORMULA	INDEX	EXPLANATION
Bromo-Benzene	C_6H_5Br	4	3 C=C 1 ring
Vinyl acetylene	C_4H_4	3	2 C=C 1 C=C
Acetonitrile	C_2H_3N	2	2 N=C
Phenyl Acetate	$C_8H_8O_2$	5	3 C=C 1 ring 1 C=O
Naphthalene	$C_{10}H_8$	7	5 C=C 2 rings

Text Frame 56: Index of Hydrogen Deficiency Examples

VI.A. Programming in Prolog

There are two different types of statements within the Prolog Language: facts and predicates. Facts are used to represent data or information that is used during the execution of Prolog programs. Collectively, all of the

facts and predicates are known as the rulebase, which is in fact the Prolog Program. There are several facts that are required in order to perform the Index of Hydrogen Deficiency calculation: the molecular formula and the valence of each atom. The facts representing the valence of atoms are shown in Text Frame 57. The format of facts is first the name of the fact ("valence" in this case) followed by the data. The three pieces of data for each fact in Text Frame 57 represent the atomic symbol, the valence of the atom, and a multiplication factor which will be explained in detail below. These facts will allow the program to convert the atoms found in the molecular formula into each atom's contribution to the hydrogen deficiency of the molecule. Also note that Text Frame 57 shows that any text following a semicolon in Prolog is treated as a comment.

```
(valence H 1 -1) ;Hydrogen is monovalent
(valence Cl 1 -1) ;Chlorine is monovalent
(valence Br 1 -1) ;Bromine is monovalent
(valence O 2 0)  ;Oxygen is divalent
(valence N 3 1)  ;Nitrogen is trivalent
(valence C 4 2)  ;Carbon is tetravalent
```

Text Frame 57: Prolog Facts

The other type of Prolog statement is the predicate. A predicate is similar to a subroutine in that it consists of a predicate name, variables that are passed to it, and a series of tasks to be accomplished. Text Frame 58 shows one of two predicates used to analyze the Index of Hydrogen Deficiency of molecules. The format of predicates is the name of the predicate, any variables to be used, and the

word "if", followed by one or more additional predicates that represent the tasks to be accomplished. Notice that only predicates can use variables, since the additional predicates after the "if" specify how values for the variables are to be obtained (see Text Frame 58). Predicate "Index_H_D" has one variable that is passed to it, the molecular formula. A molecular formula, $C_8H_9NO_2$ (for phenylglycine for example), will serve as input for the predicate. The format of the input for this example will be to convert the molecular formula into a list of atoms and the numbers of atoms found in the molecule: ((C 8) (H 9) (N 1) (O 2)). Figure 12 shows some very unusual structures for this molecular formula that are correctly addressed by this procedure. Note that although several of these structures may be un-synthesized, Equation (17) remains valid.

```
((Index_H_D ?list) if
        (count_num ?list ?count)
        (:= ?ans [/ [+ ?count 2] 2])
        (print "     The answer is " ?ans "."))
  )

          Executing Prolog Predicate
?-
?- Index_H_D ((C 8) (H 9) (N 1) (O 2))
     The answer is 5.   YES
?-
```

Text Frame 58: Prolog Predicates

Text Frame 58 also demonstrates the execution of the predicate "Index_H_D". In order to execute a Prolog predicate, type out the predicate name and values for any variable required by the predicate. Predicate "Index_H_D" requires the molecular formula in the format ((C 8) (H 9) (N 1) (O 2)) as input. Note that the order of the atoms within the list is not important. Executing a Prolog predicate is

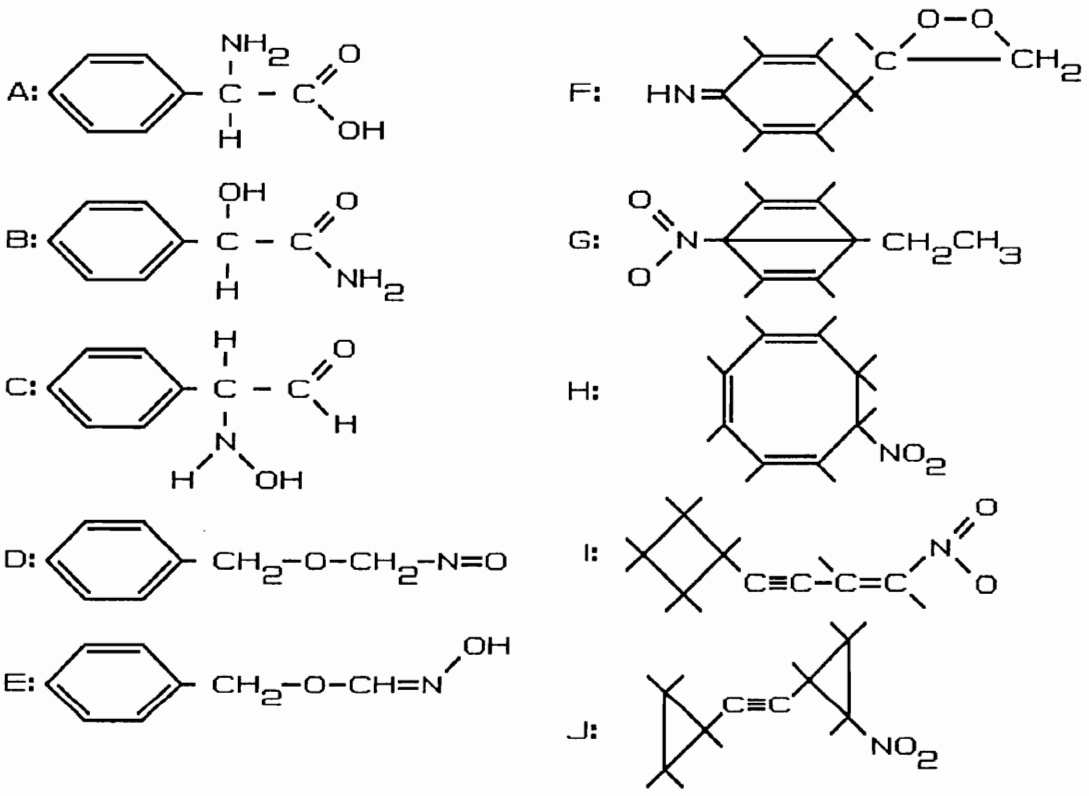


Figure 12: Structures for Molecular Formula ($C_8H_9NO_2$)

referred to as "querying" the rulebase. Prolog is attempting to determine if there is a set of values that will satisfy any variable in the predicate being executed, as well as any variable in predicates that are referenced during execution.

One could also query the facts within the rulebase shown in Text Frame 57. Several such queries are shown in Text Frame 59. Remember that any query is trying to find a set of values to satisfy the variables in the query. Query 1 in Text Frame 59 is querying the rulebase for any facts about carbon atoms. The two question marks are treated as variables and mean that any values found in these two positions will satisfy the query. Prolog prints out the fact "(valence C 4 2)" as the answer, and then responds "YES" since it was able to satisfy the query. Query 2 is searching the rulebase for any monovalent atoms. After successfully completing the query and printing the response "(valence C 4 2)", note that the prompt "?.." is different than before, and that "YES" is not printed out. This means that there are other responses that also satisfy the query. The two additional responses can be printed out by pressing

```
      Query 1
?- valence C ? ?
 YES
 (valence C 4 2)
 No More Answers
?-
      Query 3
?- valence ? ? 0
 YES
 (valence O 2 0)
?..
 No More Answers
?-
```

```
      Query 2
?- valence ? 1 ?
 YES
 (valence H 1 -1)
?..
 (valence Cl 1 -1)
?..
 (valence Br 1 -1)
?..
 No More Answers
?-
```

Text Frame 59: Prolog Queries

the "RETURN" key until "YES" is printed along with the "?-" prompt. Query 3 is looking for any atoms in the rulebase that have no effect on the hydrogen deficiency of the molecule (i.e. any atom with a multiplication factor of zero). The response is oxygen has no effect, followed by "YES" indicating the query was satisfied.

Notice that Prolog responds with "YES" at the end of the query shown in Text Frame 58. "YES" means that Prolog was able to find a set of values that satisfied all of the variables in the three predicates called by "Index_H_D". In order to understand the three predicates that make up "Index_H_D", the calculation itself must be delineated.

Equation (17) can be expressed as 1) taking twice the number of carbon atoms, 2) adding the number of nitrogens, 3) subtracting the number of hydrogen atoms, 4) adding 2, and 5) dividing the result in half. Multiplying the number of atoms by each atom's multiplication factor (shown in Text Frame 57) results in determining each individual atom's contribution toward the hydrogen deficiency of the molecule. Summing them will yield the hydrogen deficiency of the entire molecule, and Equation (17) becomes Equation (18):

$$\frac{\left[\sum_{allatoms} (Number of atoms * M factor)\right] + 2}{2} \qquad \textbf{(18)}$$

Points 4 and 5 above are easily seen in Equation (18). However, the remaining three points above require some discussion.

The goal is to be able to calculate each individual atom's contribution to the hydrogen deficiency. The multiplication factor for each carbon is 2, as indicated by Equation (17). Each nitrogen contributes one to the hydrogen deficiency of the molecule, so its multiplication factor is one. Oxygen has a multiplication factor of zero, since the number of oxygen atoms does not effect the hydrogen deficiency of molecules. Hydrogens and halogens have a multiplication factor of -1, since they are to be subtracted from the sum of contributions from carbons and nitrogens. Text Frame 57 indicates the appropriate multiplication factors.

Returning to the discussion of the three predicates that make up the predicate "Index_H_D" in Text Frame 58, the first is a call to the predicate "count_num". Predicate "count_num" returns the sum of each of the individual atom's contribution to the hydrogen deficiency as explained above. The next statement adds two to that number, then divides it by two (see Equation (18)), and assigns the result to the variable ?ans. The final statement prints out the results. Text Frame 58 also prints out the dialog resulting from a query using the predicate "Index_H_D".

Predicate "count_num" is shown in Text Frame 60. Note that there are two instances of the predicate "count_num" which are not permitted in other procedural languages. The methodology used by "count_num" to calculate the hydrogen deficiency takes the first atom from the molecular formula passed to it, calculates that atoms contribution, and recursively calls itself to process the rest of the atoms in the molecular formula. Remember that the molecular formula passed to predicate "count_num" is a list of data pairs consisting of the atomic symbol and the number of atoms found within the molecule (see Text Frame 58). The variables ?atom and ?number next to the predicate name match the first data pair in the molecular formula passed to predicate "count_num". The ".", called the "dot" operator, splits the first data pair from the list, leaving the remaining data pairs in the variable ?rst.

```
((count_num ((?atom ?number) . ?rst) ?cnt) if
        (valence ?atom ? ?mult_fact)
        (:= ?cnt2 [* ?number ?mult_fact])
        (count_num ?rst ?cnt3)
        (:= ?cnt [+ ?cnt2 ?cnt3])
  )

((count_num nil 0))
```

Text Frame 60: Predicate "count_num"

The first predicate in "count_num" is a call to the fact "valence" (see Text Frame 57 for fact "valence"). The variable ?atom has been determined from the molecular formula, and hence this predicate determines the multiplication factor (?mult_f) for the atom ?atom. Note

that in deriving Equation (18) from Equation (17), the valences of each atom type has been incorporated into the multiplication factor. The lone question mark "?" serves as a place holder for the atoms valence. These values are used to calculate the hydrogen deficiency for the atom ?atom in the next predicate by taking the product of the number of atoms and the multiplication factor. The following predicate is a recursive call to the predicate "count_num" with the rest of the molecular formula ?rst. The hydrogen deficiency for the atoms in ?rst is returned in ?cnt3. This value is then added to ?cnt2 (the hydrogen deficiency for the atom ?atom) and returned in the variable ?cnt.

Note that the recursive nature of predicate "count_num" breaks down the list of atom-number data pairs. Following the execution of predicate "count_num" as it recursively calls itself with the last data pair will demonstrate the need for the predicate's second instance in Text Frame 60. Variable ?atom will contain the last atom, ?number will contain the number of that type of atom, and ?rst will contain "nil" because there is no more data pairs in the list. "nil" is the null set and is the last member of any Prolog set. Predicate "count_num" recursively calls itself with "nil" in ?rst as the molecular formula. When executing the first instance of predicate "count_num", it fails because "nil" cannot be split by ".", the dot operator. The

second instance of predicate "count_num" is attempted since the first instance of "count_num fails. This instance succeeds since nil matches for the molecular formula variable. This instance also initializes the molecular formula count to zero. This allows the hydrogen deficiency for each type of atom to be totalled as the predicate recurses out.

VI.B. Automatic Backtracking

Query 3 in Text Frame 59 showed that with one query, all of the matching facts can be presented. When Prolog prints out "?.." as a prompt, there are other possible answers to the current query other than the first one that is presented. This is possible because of Prolog's use of choice points, which allows this automatic backtracking.

Remember that Prolog is only attempting to find values for variables in order to satisfy the query. The query is looking for all atoms that have a valency of one. When Prolog finds a match for the query as it searches through the list of "valence" facts (shown in Text Frame 57), it puts a choice point marking the point where it stopped searching. That way, when the user requests the additional answers, all Prolog needs to do is to return to the choice point and continue its search through the "valence" facts. In essence, the choice points provide a mechanism for Prolog

to continue its search through the rulebase from where it left off the last time.

VI.C. Relentless Searching

Relentless searching can be explained as a combination of Prolog's attempt to satisfy the current query (by finding valid values for all of the variables) with the use of choice points to automatically backtrack when an invalid value is uncovered. Although the Index of Hydrogen Deficiency example describes the basics of how choice points result in automatic backtracking, it does not adequately describe how the use of choice points leads to relentless searches. To further illustrate this topic, Text Frame 61 contains the dialog resulting from a functional group query on the molecule phenylglycine as well as the Prolog code used by the query (see Text Frame 2 for phenylglycine's MOL file).

Prolog Dialog:

```
?-
?- CdO PHENYLGLYCINE
Looking for a C=O in PHENYLGLYCINE:
  Found Oxygen at atom 8.  Is it a C=O?      -> Choice Point #1
  Found Oxygen at atom 9.  Is it a C=O?
*    Found C=O bond in PHENYLGLYCINE at atom 7.
   YES
?-
```

Prolog Predicates:

```
((CdO ?name) if
   (has_CdO ?name ?A1 ?A2)
   (println "*    Found C=O bond in " ?name " at atom " ?A1 "."))
((has_CdO ?name ?A1 ?A2) if
   (molecule ?name ? ?atlist ?bdlist)
     (println "Looking for a C=O in " ?name ":")
   (is_atom ?atlist "O" ?A2)                  <- Choice Point #1
     (println "  Found Oxygen at atom " ?A2 ".  Is it a C=O?")
   (finddb ?bdlist ?A1 ?A2)
   (is_atom ?atlist "C" ?A1)
  )
```

Text Frame 61: Relentless Searching

Looking at the dialog portion of Text Frame 61, the first line starts the query looking for a carbonyl group (a C-double-bond-O). Predicate "CdO" simply calls predicate "has_CdO" with the name of the molecule in ?name and then prints out the results obtained from "has_CdO". Variable ?A1 returned from "has_CdO" contains the atom number of the carbon atom of the carbonyl group, and ?A2 contains the atom number of the oxygen atom (see Table III and Table V in Chapter III).

Predicate "has_CdO" consists of four main predicates and two print statements (see Prolog Predicates in Text Frame 61). The first predicate is a call to the Molecule KB, searching for the atom list (?atlist) and bond list (?bdlist) of the molecule phenylglycine. Once the lists are found and the printing statement finishes, predicate "is_atom" searches for an oxygen atom in the list of atoms in phenylglycine. The dialog indicates that an oxygen atom (atom 8) was found. At that point, Prolog places a choice point in the atom list because the entire atom list was not searched. The next predicate call is to "finddb" which searches through the bond list for a double bond with the oxygen atom that was just found (?A2 which is atom 8). This search through the bond list fails, since in fact oxygen 8 is part of a hydroxyl group.

When the search fails, Prolog goes back to the last choice point to see if there is another oxygen atom in the atom list. Prolog is, in effect, relentlessly searching for a set of values that will satisfy the query, i.e. an oxygen for which a double bond exists. In fact there is another oxygen atom, atom 9 (see Prolog Dialog in Text Frame 61). After printing a statement that another oxygen atom has been found, predicate "finddb" is again called to search for a double bond with oxygen atom 9 (as ?A2). This time a double bond is found between atom 9 (?A2) and atom 7 (?A1). Atom 7 must be tested to make sure it is a carbon atom, which is accomplished with a call to predicate "is_atom". Since predicate "has_CdO" has succeeded, predicate "CdO" prints out the results.

It is this relentless searching that is used to do the similarity searching (refer to Chapter V.C). This ensures that in a comparison of fragments between a Target molecule and a Sim-List molecule, all possible permutations will be attempted and the permutation with the lowest total difference measure will be obtained.

VI.D. Strategy for Loading Prolog Rulebases

Prolog must load all of the facts and predicates each time CHIRULE is to be utilized. The collection of facts and predicates is known as a rulebase, and loading them is

referred to as asserting the rules into Prolog's rulebase. The approximately 3000 lines of Prolog code and the knowledge bases that make up CHIRULE can take several minutes to load. When changes must be made to a particular predicate, all instances of the predicate must be removed from the rulebase (called abolishing the rules) and then re-asserted.

The development of CHIRULE proved that abolishing then asserting individual predicates during code development was untenable. In order to ease the development process, the code for CHIRULE was broken down into 8 separate modules. This way, whenever changes need to be made, only the predicates from one module need to be abolished and re-asserted. In order to expedite this process, the first predicate in each module is defined to abolish and re-assert of all predicates found within that module. Taking module SPARC.HC as an example, the first predicate, "asparc", abolishes each of predicates contained in that module. The last thing that "asparc" does is to load the new version of the predicates in the module SPARC.HC, using the command "consult SPARC.HC".

Defining the first predicate in this fashion allows for changes to be made to the Prolog code in any one particular module without having to re-load all of CHIRULE's facts and predicates. The result is a development time savings since

the entire rulebase does not have to be loaded. The next chapter goes into further detail by explaining how this predicate definition fits into the overall development strategy for CHIRULE.

VI.E. Ring Determination Within Molecules

Searching for ring structures within a molecule is a time consuming process. The connectivity matrix represented in the MOL file must be searched one atom at a time (refer to Text Frame 2 for the MOL file for the molecule Phenylglycine). If one could calculate the presence and the number of rings found within a molecule based solely on molecular structure, a great savings in search times could be realized. The Index of Hydrogen Deficiency calculated using Equation (17), represents the number of rings, double, and triple bonds the molecule contains. Since the number of double and triple bonds within a molecule can easily be determined from a molecule's MOL file, rearranging Equation (17) enables the direct calculation of number of rings contained within a molecule. The result is shown in Equation (19):

$$\frac{(2 * TetraV + 2 + TriV) - MonoV}{2} - DoubleB - (2 * TrippleB) = Rings \quad (19)$$

The results generated by Equation (19) are used to drive the searching for rings within molecules.

The methodology described for calculating the Index of Hydrogen Deficiency uses a molecule's molecular formula. CHESS uses the molecule's MOL file to calculate the Index rather than it's molecular formula, since it is not possible to determine the number of double and triple bonds within a molecule solely from its molecular formula.

```
?-
?- num_rings IHD-A
Looking at molecule IHD-A....
   Molecule ss has 1 ring(s).
   NO
?-
?- num_rings IHD-B
Looking at molecule IHD-B....
   Molecule ss has 1 ring(s).
   NO
?-
?- num_rings IHD-C
Looking at molecule IHD-C....
   Molecule ss has 1 ring(s).
   NO
?-
?- num_rings IHD-D
Looking at molecule IHD-D....
   Molecule ss has 1 ring(s).
   NO
?-
?- num_rings IHD-E
Looking at molecule IHD-E....
   Molecule ss has 1 ring(s).
   NO
?-
```

```
?-
?- num_rings IHD-F
Looking at molecule IHD-F....
   Molecule ss has 2 ring(s).
   NO
?-
?- num_rings IHD-G
Looking at molecule IHD-G....
   Molecule ss has 2 ring(s).
   NO
?-
?- num_rings IHD-H
Looking at molecule IHD-H....
   Molecule ss has 1 ring(s).
   NO
?-
?- num_rings IHD-I
Looking at molecule IHD-I....
   Molecule ss has 1 ring(s).
   NO
?-
?- num_rings IHD-J
Looking at molecule IHD-J....
   Molecule ss has 2 ring(s).
   NO
?-
```

Text Frame 62: Calculating the Number of Rings Within a Molecule

This requires a number of changes in the original implementation of the predicates to calculate the Index of Hydrogen Deficiency, which was designed to demonstrate some of the basics behind the Prolog programming language. The new methodology counts the number of each type of atom present in the MOL file, then counts the number of double and triple bonds in the MOL file, and uses these figures to calculate the Index. The results of applying this procedure

to all of the structures shown in Figure 12 are shown in Text Frame 62. The number of rings in all of the structures, even the unusual ones, are correctly predicted. Refer to module CHEM.HC in Appendix F for "num_rings", "cnt_rings", and several additional predicates which implement Equation (19).

VII. System Overview

The work described below develops the concept of a general purpose shell for creation of chemistry based expert systems. This shell contains a molecular property evaluator, functional group recognition capabilities, an automated knowledge acquisition utility, and the capability to symbolically reason on chemical structures. The idea behind the shell is that scientists could use it to build an expert system for methods development in any specific area relevant to their research.

Each individual part of the CHESS system has been explained in previous chapters. What remains is to discuss how each of these parts interacts with and relies on each other. The information contained in this chapter covers the details of how the entire CHESS system is put together. Sections VII.A through VII.C are likely to be of interest only to those who wish to understand the processes needed to integrate packages together. Section VII.D should be of interest to anyone wanting to implement systems using CHESS, as it demonstrates examples of users interactions with CHESS, OPTIMER, and CHIRULE.

VII.A. System Technical Notes

This section examines the specifics of the computing facilities used, the software development packages and configurations employed, and information on software that was specifically written for this work. It also includes directions on how to start up each of the software packages and any appropriate notes on running them.

VII.A.1. Computer Workstation

The unifying computer for this project is a personal computer workstation donated by the Intel Academic Relations group[126]. The Intel 8 slot tower system unit has a 33mHz i386 processor, 8 meg of memory on the system board, 6 additional meg of memory on an expansion board, 150 meg SCSI hard disk and tape backup unit, 3COM 505 16 bit Ethernet card, Paradise 1024 VGA video card, and an NEC MultiSync 3D VGA monitor. QuarterDeck's expanded memory manager, QEMM v6.0, was used to load all device drivers into high memory, leaving 576Kb of main memory free after bootup with DOS v4.01.

QuarterDeck's DESQview v2.4 was used to run six or more concurrent DOS sessions at a single time, each with as much as 564K of conventional memory available to it. Expanded memory can also be assigned to each session, as was required by SPARC. DESQview was chosen over Microsoft Windows

because all of the applications used were DOS based applications, and it is believed that DESQview handles task switching DOS-based applications in a superior manner. DOS 4.01 was used because DOS 5.0 required significantly more memory to run[127].

VII.A.2. SPARC

SPARC is the molecular property estimator described in Chapter III.C. SPARC and all of its associated programs run on the PC workstation under DESQview. The computer workstation's 8 meg of memory was not sufficient to run SPARC under DESQview. Although SPARC installation instructions indicated that 8 meg would suffice, the loading of all of the required device drivers required the purchase of an additional memory board. A DOS extender from ALS Prolog, provided with SPARC, is utilized to gain access to all of the memory. SPARC provides an integrated environment to gain access to the property evaluator, a graphical molecular structure drawing tool (MEDIT), a SMILES string generator (CONVB), and a batch mode interpreter for performing property estimations based on information read in from a file.

Once installed, running SPARC is accomplished by executing the batch file "GOSPARC.BAT" which insures that an ANSI driver is loaded before starting SPARC. There are two

parameters required to run SPARC: 1) "-h2048" means reserve 2 megs of memory for heap storage, and 2) "-s1024" means reserve 1 meg of memory for the stack. This is part of the reason that SPARC requires 8 megs of memory to run. Upon starting up SPARC, the user is presented with a menu of possible tasks, including analyzing a molecule, exiting SPARC, and starting the batch mode program.

SPARC has the capability to run as an integrated environment. If SPARC were being used in this fashion, the user would select to run a molecule. MEDIT would then be used to graphically draw the molecular structure. When the structure was complete, the user would select SMILES STRING from the MEDIT menu, and CONVB would be called to generate the SMILES string which would be passed back to SPARC. The user would then be presented with a list of properties that could be calculated, one at a time. CHESS does not make use of SPARC directly. Instead, several of SPARCs programs are utilize by themselves. The development environment used to incorporate SPARCs programs into CHESS is described in detail in Section III.C.3. This section includes information on how information is exchanged between CHESS and SPARC.

The SMILES string generator CONVB, described in Chapter III.A, is used to generate SMILES strings for all molecules within the CHESS Molecule KB. The MOL file input to CONVB

(C:\SPARC\MEDIT\MOL\TEST.CVT shown in Text Frame 5 in Chapter III.A) is slightly different than the standard MOL file used by MDL's programs (see Text Frame 2). The MOL file format used by MEDIT is fashioned after that used by the chemical drawing package Alchemy from Tripos Associates[128]. CONVB uses a subset of this format (see Text Frame 4), deleting all hydrogens within the molecule. This format is very similar to MDL's format, and CHESS handles the necessary conversions. The output of CONVB is a file (C:\SPARC\SS\TEST.SS) containing the SMILES string for the molecule.

The batch mode interpreter, described in Chapter III.C, is used by CHESS to gain access to the property estimation facilities. SPARC inputs a file (C:\SPARC\BATCH.DAT, see Text Frame 8) consisting of the SMILES string of the molecule, default conditions, and an abbreviation for the property to be calculated. The output file (C:\SPARC\BATCH.OUT, see Text Frame 9) contains the SMILES string of the molecule, property name, conditions, default values, and the calculated property value. The user can request the creation of a "dribble" file (C:\SPARC\BATCH.DRB, see Text Frame 10) where intermediate results are shown. CHESS writes the SPARC input files and then reads the SPARC output files in order to obtain values for the various molecular properties. The transferring of

files between the programs is described in detail in Section III.C.3 where CHIRULEs development environment is discussed.

VII.A.3. MACCS and ChemTalk

MACCS[15] is the package used to graphically input the molecules into CHESS by way of saving the structures to MOL files (see Chapter III.A). MACCS v5.1x runs on VTCC1, the University's VAX 8800, running VMS 5.4-3. Another MDL product, ChemTalk, emulates a Tektronix 4105 color graphics terminal to gain access to the MACCS software. ChemTalk v1.40 uses a serial connection to an IBM ROLM Dataline connection, which is connected electronically to VTCC1 over a DEC LAT terminal server.

Setting up CPSS and ChemTalk is accomplished as described in the installation manual. Running ChemTalk is accomplished by first adding the directory name to the current path (path=%path%;C:\CPSS14), setting an environment variable (set cpss=C:\CPSS14), running an initialization procedure (C:\CPSS14\INITCPSS.COM), and then running the CPSS program (C:\CPSS14\CPSS14 CTALK). Note that although this command specifies running ChemTalk, ChemBase is still accessible. Running CPSS under DESQview requires no special device drivers.

There are three places where various setup parameters must be changed to enable ChemTalk to run properly with the

MACCS software. From the ChemTalk Main Menu, choose settings to change Com Port (use COM1) and other serial port parameters (baud rate, data bits, parity, etc.). Since the other two settings are related to the terminal emulation, go to the ChemTalk Main Menu, and choose Terminal from the menu bar. This will bring up a character-based screen that enables the user to log on to the IBM Rolm system, and then onto the VAX. Typing "alt-e" will bring up the Emulator Settings Menu that allows changing both the graphics and text terminal emulations, as well as specifying the keyboard type and character set. The graphics emulation that will be used in this work is the Tektronix 4105 color graphics terminal, and the text terminal emulation will be a VT220. Lastly, typing "alt-s" will bring up the Terminal Settings menu where default settings for the functions of the backspace key or the EOL string can be changed.

```
WELCOME TO THE VIRGINIA TECH CBX DATA NETWORK

CALL, DISPLAY OR MODIFY?
call vtcc1
CALLING 42009
CALL COMPLETE

DECserver 500 Terminal Server  V1.0 - LAT V5.1

Please type HELP if you need assistance

Local> connect vtcc1
Local -010- Session 1 to VTCC1 established

TechCluster Node - VTCC1

Username: MDL
Password: XXXXXXXX
        Welcome to VAX/VMS version V5.5-2 on node VTCC1
    Last interactive login on Wednesday, 24-FEB-1993 16:00
    Last non-interactive login on Tuesday,  2-FEB-1993 00:01
                Virginia Tech Computing Center

* * * * Enter NEWS to read old messages.  Use the 4HELP command
        to report problems; send suggestions to userid SUGGEST.
$
```

Text Frame 63: Logging onto VTCC1 using the IBM Rolm System

After setting up the terminal emulation as described above, the next step is to log onto the VAX. After selecting Terminal from the ChemTalk menu bar, hit return several times. This will bring up the IBM ROLM CBX Data network (Text Frame 63). Typing "call VTCC1" will connect the user with the DEC Terminal Server. From this point, typing "connect VTCC1" will connect the user to VTCC1, Virginia Tech's DEC VAX 8800. Log onto the computer account by providing the Username and Password.

The setup for MACCS is handled by the system administrator. After logging onto VTCC1 (see Text Frame 63), typing "MACCS" will start up the MACCS system. The user will be presented with the MACCS login screen, and then must specify the search database and the terminal type utilized (see Text Frame 64). Typing "aldrich:", then a return for

```
$ MACCS

                    ********************************

                                MACCS

                    ********************************

                            Revision 5.1x

            This program contains unpublished confidential,
                 proprietary and trade secret material
                 belonging to Molecular Design Limited

                Copyright (c) Molecular Design Limited
                1984, 1985, 1986.  All Rights Reserved.

                            Good Evening

Enter name of database: ALDRICH:
Enter password:
   **** DEMONSTRATION DATABASE DERIVED FROM ALDRICH CATALOG ****
Please choose graphics device
 or <CR> for nongraphics, L for list: L

 1. Imlac              2. Tektronix 4114     3. VT-640
 4. Tektronix 4105     5. Lundy 568x         6. DQ650M
 7. HP 2623A           8. GT40               9. Envision
10. Tektronix 4107    11. Envision II       12. ChemTalk Terminal

Please choose graphics device
 or <CR> for nongraphics, L for list: 4
              32041 ENTRIES IN DATA BASE
```

Text Frame 64: MACCS Logon Procedure

the password (i.e. no password), and then typing 4 to indicate a Tektronix 4105 color graphics terminal, will allow the user to begin to use the MACCS system.

VII.A.4. KEY

KEY was developed under a grant from the Proctor & Gamble company, who required a DOS based application. It is written in Microsoft C v4.0, and is currently at revision 6. Key consists of approximately 5,000 lines of C code, broken into five modules. Microsoft C runs on the computer workstation. A separate DESQview window is used to compile and link all of the modules. Microsoft CodeView is used as a symbolic debugger, enabling errors to be traced by watching any of the programs variables while execution of the program proceeds. No special setup information is required for either Microsoft C or Codeview.

The five modules, GRIDEFS6.H, KEY6.C, GRID6.C, ANALY6.C, and NEW6.C (see Appendix C for complete code listings), each cover a different task that KEY performs. GRIDEFS6.H is a C "include file" that contains all data structures used to represent the knowledge grid, as well as constants used by KEY. In addition, it contains definitions for each and every C subroutine, grouped into the modules where they reside. While C does not require this, it will

prevent a developer from accidentally switching arguments or passing an incorrect variable type for all the subroutines.

KEY6.C contains the subroutines that elicit and create the trait and element lists from the user, as well as the menuing routines. GRID6.C handles generating, editing, and saving the knowledge grid. Grid analysis subroutines, including reporting of the analysis results, are included in module ANALY6.C. The last module, NEW6.C, is where the rulebase generation subroutines are grouped. Each of these modules are individually compiled to object files, which are then linked together into one executable file (KEY6.EXE). Breaking the code into separate modules saves development time. When corrections are made, only that one module must be re-compiled. Linking the object modules together is a relatively quick process.

There are no special instructions needed to install KEY, which consists only of the executable program "KEY6.EXE". After changing to the directory that contains KEY6.EXE, typing "KEY6" will begin the grid acquisition as described in Chapter IV.C. All element, trait, or grid files to be used must be in the same directory as KEY6.EXE. In addition, if a Turbo-Prolog rulebase is to be generated, the file "T-Prolog.SAV" must be located in the same directory. This file contains the inference engine code to run the rules generated from the knowledge grid.

The output of KEY is a rulebase that can be run on one of two expert system inference engines (see Chapter IV.E, Section VII.D.2 below, and Appendix E). The first was written as part of CHESS in VPI Prolog, and has already been discussed. The second was written in Turbo-Prolog v2.0 from Borland International and runs on a DOS based PC.

VII.A.5. Turbo-Prolog and KEY

Setting up Turbo-Prolog is discussed in the installation section of the Users Manual. There are three compiler directives that had to be changed in order to allow Turbo-Prolog to run under DESQview: StackSize, Code, and Screen BIOS. To eliminate the error "1003 Heap Overflow" when trying to compile and run a program, place "Code=2048" at the beginning of the program to increase code space and set the stack size (under Options /Compiler Directives /Memory Allocation) to 1024. The user must also instruct the screen display routines to use the BIOS calls to enable DESQview to switch between windows. This is accomplished by adding "-X" to the Turbo-Prolog command line (i.e. "Prolog -X").

The rulebases generated for the VPI Prolog inference engine and the Turbo-Prolog inference engine are quite different. The VPI Prolog rulebase consists mainly of a list of all of the rules resulting from the completed

knowledge grid. The Turbo-Prolog rulebase consists of these rules, but also contains the Prolog code necessary to drive the inference engine, which consists of approximately 200 lines of Prolog code broken into three sections. The first section contains all structure definitions, a characteristic of procedural languages such as Pascal or "C". Turbo-Prolog requires these structure definitions. Most Prologs running on larger computer systems do not constrain programmers in this fashion. Turbo-Prolog was not chosen for the implementation of CHESS. The second section contains all of the Prolog predicates (called "clauses" by Turbo-Prolog). The third section contains the rules generated from the knowledge grid.

When KEY generates a Turbo-Prolog rulebase, it first checks for, or opens the file "T-Prolog.SAV" which contains all of the inference engine predicates (sections one and two described above). It then transfers this information to the rulebase file, and appends all of the rules from the knowledge grid. The Turbo-Prolog inference engine implements the combined implication strengths (Equation (5) on page 137) and certainty factor algorithm (Equation (6) on page 139) described previously for the VPI Prolog inference engine.

To run the newly created Turbo-Prolog rulebase, first copy it to the Turbo-Prolog directory. Then change to that

directory and type "Prolog -x" to run Turbo-Prolog. Type "F" to select the files menu, and select "Load" to load in the rulebase. Then select either "Compile" to generate a stand alone executable program, or "Run" to execute the rulebase from within Turbo-Prolog.

VII.A.6. VPI Prolog and CHESS

VPI Prolog[129] was used to write CHESS, CHIRULE, and the inference engine for OPTIMER, the knowledge base generated by the automated knowledge acquisition system KEY. VPI Prolog v1.039 runs on VTCS1, the Computer Sciences MicroVAX II running VMS 5.4-3. It was accessed by running ProcommPlus v1.1B from Datastorm Technologies, Inc., under DESQview. ProcommPlus is a terminal emulation program and a VT105 emulation was used. Procomm used a serial connection to an IBM ROLM Dataline which communicated electronically to VTCS1 over CSRN, the Computer Science Research Network.

CHESS and CHIRULE consist of approximately 3,000 lines of Prolog code, broken into ten modules. After logging onto VTCS1 using ProcommPlus, the user must change to the directory where all of the Prolog code resides before starting up VPI Prolog. The code resides in the sub-directory DUA0::[STAUFFER.FGROUP], and the command "SET DEFAULT [STAUFFER.FGROUP]" changes the default directory appropriately. Prolog itself resides on VTCS1 at

$user:[roach.deighan]Prolog.exe (and on VTCC1 at dua5:[roachjw.deighan]Prolog.exe). The symbolic definition "HCC filename" (HCC :== $user:[roach.deighan]Prolog.exe) will start up Prolog and load all of the predicates contained in "filename.HC". CHESS has been written such that the master file "RES.HC" will load all of the remaining modules. Loading the rulebases and executing CHIRULE are covered in section VII.D below.

VII.B. Knowledge Bases

CHESS's chemical intelligence is stored in a number of distinct separate knowledge bases. Chapter III described knowledge bases in detail and the key differences between them and data bases. This section describes all of the knowledge bases (hereafter referred to as KBs) that are utilized by CHESS and CHIRULE (see Figure 13).

The KBs shown in Figure 13 and described in Table XI do not all serve the same functions. The four in the corners of Figure 13 are static, being stored in a disk file and read in each time CHESS is loaded to create a run-time version. The others are run-time only KBs and are not stored in disk files, being created from the static KBs while CHESS is running. Each consists of a fact name which is used to refer to the KB, followed by the various pieces of information they contain (see Chapter VI for a complete

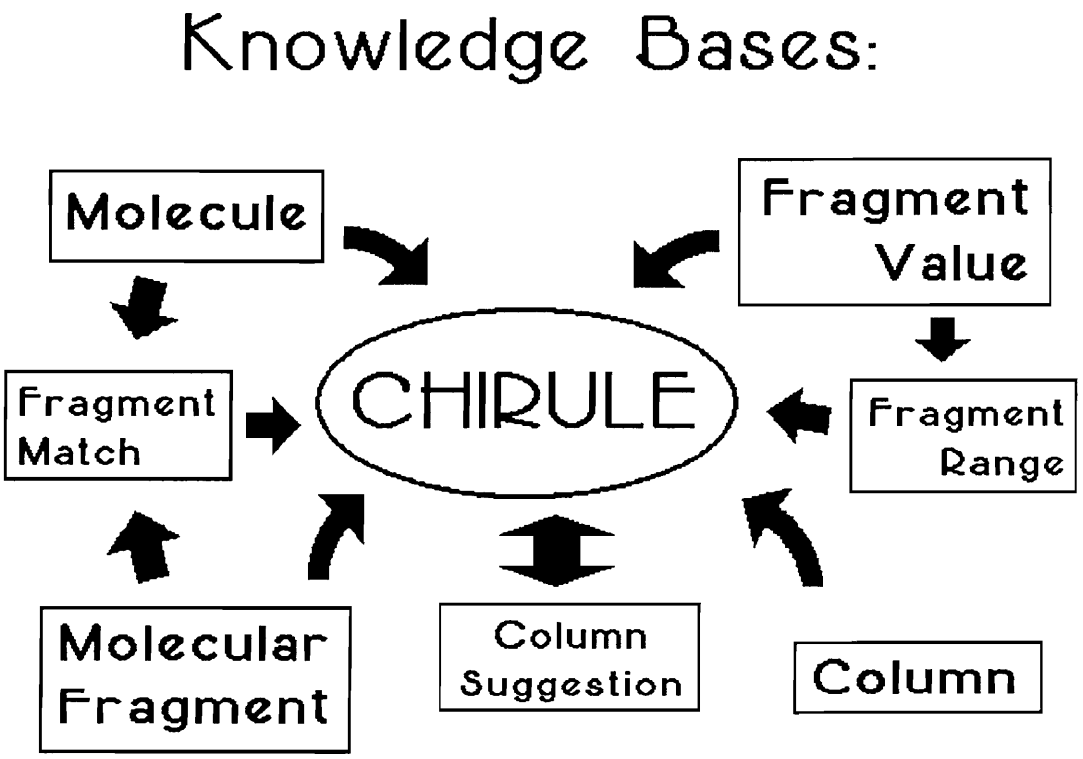


Figure 13: CHESS and CHIRULE Knowledge Bases

Table XI: CHESS and CHIRULE Knowledge Bases

MOLECULE KNOWLEDGE BASE [MOL.DB] - Contains the information read in from the mol file generated by MDL's MACCS program. It consists of the fact name "molecule", the name of the molecule, the SMILES string, an atom list and a bond list.
Example: ((molecule molecule-name smiles-string atom-list bond-list))

MOLECULE FRAGMENT KNOWLEDGE BASE [MOL-FRAG.DB] - Contains the Smiles string for the entire chiral molecule, and its 4 fragments generated by splitting the molecule up around the chiral center. It consists of the fact name "molfrag", the Smiles string of the molecule, and the Smiles strings of the 4 fragments in order of decreasing number of atoms contained by each.
Example: ((molfrag MOLSS fgmt1ss fgmt2ss fgmt3ss fgmt4ss))

FRAGMENT VALUE KNOWLEDGE BASES [FV-POL.DB, FV-ALPHA.DB, FV-BETA.DB] - Contains the values generated by SPARC for the property's listed for each fragment. It consists of the fact name "fg-valu", the Smiles string of the fragment, the property abbreviation used by SPARC (pol, alpha, beta, vol), and the property values for the fragment.
Example: ((fg-valu FRAGSS property fgmtvalu))

CHIRAL COLUMN KNOWLEDGE BASE [CHIRLCOL.DB] - Consists of the fact name "chiral-col", the smiles string for the molecule, the column or columns used to separate the enantiomers, and the column reference.
Example: ((chiral MOLSS chiral-column spectra-reference))

FRAGMENT VALUE RANGE KNOWLEDGE BASE [runtime only] - Consists of the fact name "fvalu-range", the fragment smiles string, the name of the property (pol, alpha, beta, vol), the delta value used to generate the value, and the value itself. The purpose is to place each fragment into boxes, so that they can be compared to other fragments to come up with "similar" fragments. Two entries in "fvalu-range" are generated for each value in the fragment database "fg-valu". Generated by the predicate (gen_fvrange_db ?property ?delta) from the fragment value database "fg-valu".
Example: ((fvalu-range ?fragment ?property ?delta ?value))

FRAGMENT MATCH KNOWLEDGE BASE [runtime only] - Consists of the fact name "fragmatch", the Smiles String of the molecule of interest and the matching molecule, the number of fragments that were matched, the fragment in the molecule of interest and its match in the matching molecule, the property used to match the fragments, the delta value used, and the actual difference value.
Example: ((fragmatch ?ssa ?ssb ?nmatch ?frag ?frag2 ?property ?delta ?diff))

COLUMN SUGGESTION KNOWLEDGE BASE [runtime only] - Consists of the fact name "column-sugg", the Smiles String of the molecule of interest, the Smiles String of the molecule that is considered to be a "match", and a numerical value representing the strength of the match.
Example: ((column-sugg ?ssa ?ssb ?value))

description of Prolog facts and predicates). This section also covers how each of them was created, the purpose they serve, and how they interact with each other. Appendix G contains complete source listings for each of the static KBs used in this project. The file RES.HC in Appendix F also contains descriptions of the static KBs.

The main knowledge base is the Molecule KB, consisting of the fact name "molecule", followed by the molecules name, its SMILES string, and its atom and bond lists. The static version of the Molecule KB exists in the file "MOL.KB", where each molecule is listed one after another in the format shown in Text Frame 64. The predicate "load_mol_kb" reads in this file, creating the run-time Molecule KB, and "save_mol_kb" takes the current run-time KB and saves it to the file "MOL.KB". The predicate "load_mol" reads in a MOL file, adding it to the run-time Molecule KB, and "save_mol ?name" adds the molecule ?name to the static Molecule KB. When a new molecule is read in using "load_mol", it must be saved

```
((molecule PHENYLGLYCINE "c(ccc1C(N)C(=O)O)cc1"
   (     (atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0)
         (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 O 0)
         (atom 9 O 0) (atom 10 C 3) (atom 11 N 0) )
   (     (bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2)
         (bond 5 6 1) (bond 6 1 2) (bond 7 8 2) (bond 7 9 1)
         (bond 7 10 1) (bond 10 4 1) (bond 10 11 1) )
 ))
((molecule MANDELAMIDE "c(ccc1C(O)C(=O)N)cc1"
   (     (atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0)
         (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 O 0)
         (atom 9 C 0) (atom 10 O 0) (atom 11 N 0) )
   (     (bond 2 1 2) (bond 3 2 1) (bond 4 5 1) (bond 5 6 2)
         (bond 6 1 1) (bond 4 3 2) (bond 7 3 1) (bond 8 7 1)
         (bond 7 9 1) (bond 9 10 2) (bond 9 11 1) )
 ))
((molecule "METHYL MANDELATE" "c(ccc1C(O)C(=O)OC)cc1"
      .
      .
      .
```

Text Frame 65: Molecule Knowledge Base

to the static KB if it is to be retained the next time CHESS is used.

When CHIRULE fragments the chiral molecule around the chiral carbon creating four fragments, the fragments' SMILES strings are stored for use by the similarity searching algorithm. The Molecular Fragment KB contains this information. It consists of the fact name "molfrag", the SMILES string of the molecule, and the SMILES strings of each of the four fragments. The static version of this KB is stored in the file "MOL-FRAG.KB" in the format shown in Text Frame 66. The predicate "load_molfrag_kb" reads in the static KB from the file "MOLFRAG.KB". The run-time KB is added to by the predicate "getall_smiles" while the static KB is added to by the predicate "saveall_smiles".

```
"c(ccc1C(O)C(=O)O)cc1" "c(ccc1)cc1" O=CO 0 ""
"c(ccc1C(O)C(=O)N)cc1" "c(ccc1)cc1" O=CN 0 ""
"c(ccc1C(O)C(=O)OC)cc1" "c(ccc1)cc1" O=COC 0 ""
      .
      .
      .
```

Text Frame 66: Molecular Fragment Knowledge Base

The similarity searching algorithm uses four molecular properties for each of the fragments, and must have the values for each of the fragments for each of the properties. The Fragment Value KB contains the fact name "fg-valu", the SMILES string of the molecular fragment, the property abbreviation, and the fragments value for that property. For each of the properties, the values for the static KBs are stored in the following files: average molecular

polarizability - FV-POL.KB, molecular volume - FV-VOL.KB, electron donating capability - FV-ALPHA.KB, and electron withdrawing capability - FV-BETA.KB. Text Frame 67 shows several entries from each of these files. The predicate "load_value_kb" reads in the static KB from all of the above files. The run-time KB is added to by the predicate "getall_sparc" while the static KB is added to by the predicate "getall_sparc" which calls "savef_value".

```
FV-POL.DB                          FV-VOL.DB
"c(cccl)ccl" pol 1.03565           "c(cccl)ccl" vol 88.87
O=CO pol 0.34022                   O=CO vol 42.69137
O=CN pol 0.4066                    O=CN vol 35.62097
O=COC pol 0.52369                  O=COC vol 61.6403

FV-ALPHA.DB                        FV-BETA.DB
"c(cccl)ccl" alpha 0               "c(cccl)ccl" beta 0.12
O=CO alpha 0.5952                  O=CO beta 0.4379
O=CN alpha 0.39068                 O=CN beta 0.87117
O=COC alpha 0                      O=COC beta 0.40072
```

Text Frame 67: Fragment Value Knowledge Base

Once the similarity search yields a list of molecules similar to the target molecule, it needs to suggest a column that would serve as a good starting point for the new method development. The Chiral Column KB contains the fact name "chiral-col", the SMILES string of the molecule, the chiral column that successfully separated the molecule, and a reference of where to go to get additional information such as column conditions. The predicate "load_chiral_kb" reads in the static KB from the file "CHIRLCOL.KB", several entries of which are shown in Text Frame 68. Note that it

```
"c(ccc1C(O)C(=O)O)cc1" "Chiralpak WH" "Daicel p27"
"c(ccc1C(O)C(=O)O)cc1" "Chiralcel OD" "Daicel p27"
"c(ccc1C(O)C(=O)N)cc1" "Chiralcel CA-1" "Daicel p44"
"c(ccc1C(O)C(=O)OC)cc1" "Chiralcel OK" "Daicel p44"
```

Text Frame 68: Chiral Column Knowledge Base

can contain multiple column references for each molecule. The run-time and static KB is added to by the predicate "getinfo_chiral", which gets the chiral column information from the user.

The information in these static KBs are all stored in disk files and read in each time CHIRULE is run to generate their run-time versions. There are a few KBs that are only created at run-time and are not saved to a file. These three KBs all play a part in performing the similarity search to come up with a list of column suggestions for the target molecule.

The first of these run-time-only KBs involves placing all of the fragments in the Fragment Value KB into the appropriate bins in preparation for the similarity search. Entries in the Fragment Value Range KB take the form:

"fvalu-range ?fragment ?property ?delta ?bin"

They are generated by the predicate "gen_fvrange_db", which is called once for each property utilized in the similarity search. This is all accomplished by the predicate "gen_all_range" which takes the delta value as a parameter (refer to Chapter V.C Similarity Searching).

The delta value defines the size of the bins, effecting how many fragments will be found within each. When experimenting with various delta values, it is the users responsibility to manually abolish previous versions of the

run-time Fragment Value Range KB by typing "abolish fvalu-range" while within Prolog. Failure to do this will result in extending search times because of multiple sets of bins (each with different delta values) being present.

The Fragment Match KB "fragmatch" is the second KB that exists in a run-time version only, and its entries have the following format:

"fragmatch ?ssa ?ssb ?nmatch ?f ?f2 ?property ?delta"

This KB is at the heart of the similarity searching algorithm. It is created by the predicate "psimilar" when two fragments (?f & ?f2) are found to match between the target molecule (?ssa) and another molecule (?ssb) in the Molecule KB. The match occurred for a property (?property) at a delta value (?delta), and there are a total of ?nmatch non-identical fragments to match. Each entry is removed from the Fragment Match KB by the predicate "get_property" which calculates the total difference measure between two molecules.

The final run-time only KB is the Column Suggestion KB. It consists of the two matching molecules SMILES' strings and the total difference measure. It is created by the predicate "assert_colsugg" which is called by the predicate "calc_diff" after it finds a pair of similar molecules. After all similar molecules are found, the predicate "make_col_list" converts the Column Suggestion KB into a

list of column suggestions, removing entries as they are added to the list. This list is then ordered putting the lowest difference values first, and finally printed out.

VII.C. Network Integration and Information Flow

Establishing a development strategy for the formation of a general purpose expert system shell that understands basic tenants of chemistry was imperative to success. The development strategy for CHESS (and hence KEY, OPTIMER, and CHIRULE) relies on the computer networking capabilities that exist and defines the achievable integration. Included are the programs used in each environment, any programs used to transfer data between programs, and how the programs interacted with one another.

As stated in Chapter VII.A.1 above, QuarterDeck's DESQview running on a PC Workstation was the platform implemented to integrate the various pieces of commercial software used in the following strategies. DESQview is used to maintain separate DOS windows that are open to several different software packages concurrently. This ability to switch between windows running different software packages saves the developer a substantial amount of time. These time savings, as well as other advantages realized using the strategies, are pointed out in the following sections. Each

describes the development strategy for a different section of the project.

VII.C.1. KEY Development

The development strategy for KEY, shown in Figure 14, consists of three DESQview windows: the editor window (WordPerfect 5.0), a window for compiling programs, and a window for running CodeView. KEY is written in Microsoft C in a DOS PC environment. After making changes to one of the modules in KEY, those changes would be saved to a disk file, then compiled. Errors that are encountered during the compilation process produce warning messages that are printed on the screen. These messages could then be addressed one at a time and corrections made in the editing window.

When required, CodeView could be employed to trace down logic errors in the program. From within CodeView, a developer can place breakpoints in the code that will cause the execution of the program to halt at those points and allow the interrogation of any of the programs variables. This proves especially useful when expanding the size and capabilities in the data structure that stores the knowledge grid.

Having the ability to keep one window open with an editor in addition to another window where the actual

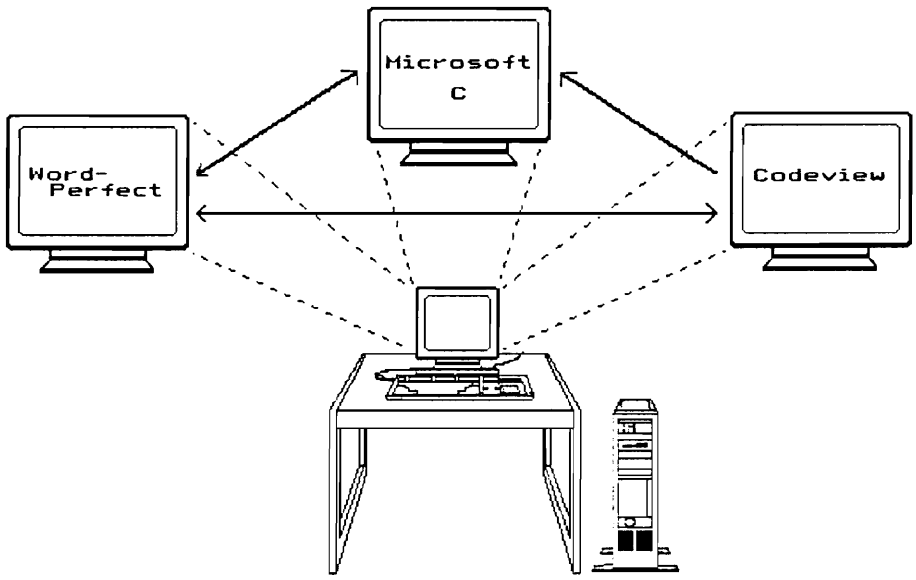


Figure 14: KEY Development Strategy

compiling of the program takes place is a vital time saving measure. It relieves developers from having to exit the editor, run the compiler, then start up the editor again to make further changes. In addition, when errors are detected by the compiler, the actual error messages can be viewed in one window while corrections are being made in another, providing additional time savings.

VII.C.2. Prolog Code Development

Prolog was used to develop a majority of the programs that make up CHIRULE. The development strategy utilized consists of three simultaneous windows running under DESQview on the computer workstation described above (see Figure 15). Those windows included an editor window, a file transfer window, and the terminal emulation window using Procomm Plus. Local editing of program files allowed the use of more flexible editors and faster processing of files. Word Perfect v5.0 was used as a text editor to create and edit all of the program modules. These modules were saved in ASCII format to the local PC workstation's hard disk, then transferred over the Ethernet backbone to the correct sub-directory on VTCS1 in the second window using the file transfer protocol (FTP). The FTP package used was PC/TCP v2.05 from FTP Software[130].

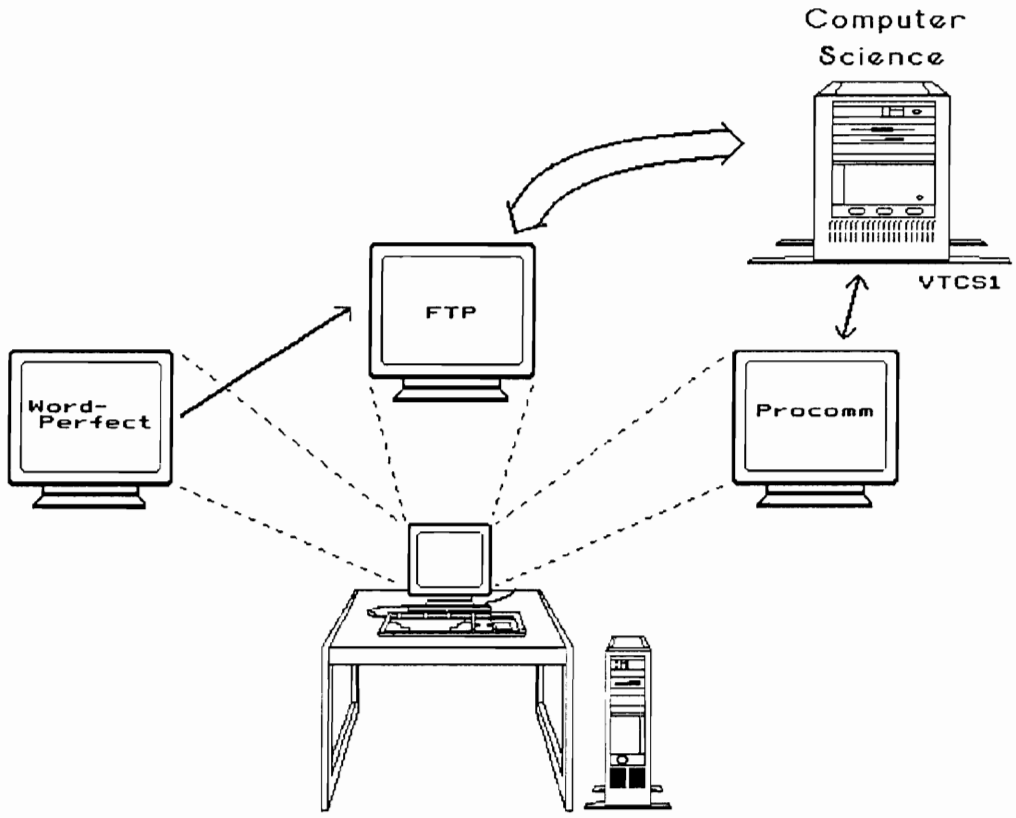


Figure 15: Prolog Development Environment

Note that all of the program modules are edited locally and transferred to VTCC1. Any corrections or additions that need to be made are done locally, and the modules are re-transmitted and reloaded into Prolog. The "ANEW" strategy explained in Chapter VI allows the reloading on only the effected modules and eliminates the time it would take to reload all of the program modules.

Ideally the file transfer window could be eliminated, and the development strategy would consist of only two windows. This could be accomplished by adding NFS (Network File Service) capability to VTCS1. Currently, VTCS1 uses a networking package called MultiNet for its Ethernet networking capabilities, providing both Telnet (terminal access) and FTP (File Transport Protocol) services. If the NFS package for MultiNet were installed, the hard disk of VTCS1 could be mounted directly by the computer work station making it appear to be simply another local hard disk. Thus files that were saved by the text editor would reside directly on VTCS1, where VPI Prolog would have direct access to them.

VII.C.3. CHIRULE Development

In running the CHIRULE system, all of the advantages of an integrated computer environment can be demonstrated. This work takes advantage of software packages that run on a

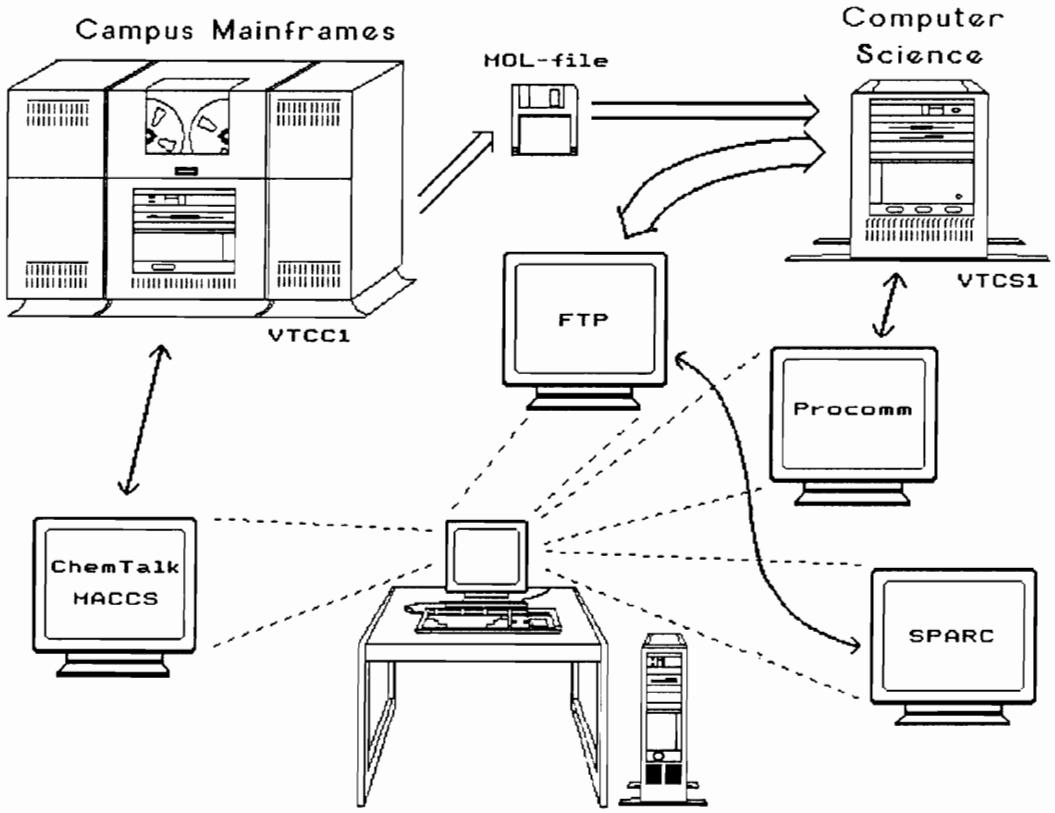


Figure 16: CHIRULE Development Environment

number of computer platforms. Each of these packages have been chosen because of their own unique characteristics. It is through the development strategy shown in Figure 16 that they all come together in one computer workstation.

CHIRULE's strategy (Figure 16) consists of programs running in four simultaneous windows: MACCS for drawing the molecule, an FTP window for transferring files, Procomm for running Prolog on VTCS1, and a window for running SPARC. ChemTalk, emulating a Tektronix 4105 color graphics terminal, is used to gain access to MACCS running on VTCC1. The FTP window is used to transfer the MOL file from VTCC1 to VTCS1. The FTP window is much more involved in this strategy because of this transfer and its responsibility for the exchange of information with SPARC, which is running locally on the computer workstation.

SPARC and the programs that come with it are used in the following ways. CONVB is used to generate the SMILES strings for not only all of the molecules within the Molecule KB, but also for all of the fragments in the Fragment Value KB. SPARC is also used to generate the various property values for each of the fragments. All three uses require that information be transferred from VTCS1 where CHESS and CHIRULE reside. These transfers are accomplished using batch files in the FTP window.

The first transfer occurs when CONVB is used to generate the SMILES string for a new molecule being added to the Molecule KB.

An entry in the Molecule KB has already been made by reading in the new molecules MOL file. In order to fill in the SMILES string for the new molecules, CHESS writes out a modified MOL file that CONVB will accept (see Text Frame 5 in Chapter III.A). This MOL file must be transferred from VTCS1 to the local computer workstation where CONVB will generate the SMILES string. The resulting SMILES string must be sent back for inclusion in the Molecule KB. The batch file that accomplishes this, DO-CONVB.BAT (shown in Text Frame 69), uses the FTP protocol to transfer the MOL file.

```
@ECHO OFF
C:
ECHO.
ECHO     Getting TEST.CVT from VTCS1.....
ECHO.
C:\NETWORK\FTP\FTP -u staufferst password VTCS1
        get [.fgroup]test.cvt c:\sparc\medit\mol\test.cvt > nul
ECHO     Running CONVB to generate the Smiles String....
ECHO.
CD C:\SPARC\MEDIT
CONVB
ECHO     Putting TEST.SS to VTCS1.....
ECHO.
C:\NETWORK\FTP\FTP -u staufferst password VTCS1
                put c:\sparc\ss\test.ss [.fgroup]test.ss > nul
CD C:\
```

Text Frame 69: DO-CONVB.BAT

After the new molecule has been input into CHIRULE, the molecule is separated into four fragments around the chiral carbon within the molecule. In order to check if these fragments reside within the Molecular Fragment KB, they must be converted into their equivalent SMILES strings. CHESS outputs a separate MOL file for each fragment These must be transferred to the local computer workstation where CONVB

will translate them into their respective SMILES strings. These must be transferred back to VTCS1 where they are added to the Molecular Fragment KB. The batch file CONVBALL.BAT, shown in Text Frame 70, accomplishes this.

```
@ECHO off
C:
ECHO.
ECHO     Obtaining Smiles strings for fragments of %1.....
ECHO.
ECHO         Getting [%1.cv?], input files for CONVB.....
cd c:\sparc\medit\mol
C:\NETWORK\FTP\FTP -u staufferst password VTCS1
                                    mget [.fgroup]%1.cv* > nul
cd c:\sparc\medit
ECHO              Running CONVB.....
ECHO.
copy c:\sparc\medit\mol\%1.cv1 c:\sparc\medit\mol\test.cvt > nul
CONVB
copy c:\sparc\ss\test.ss c:\sparc\ss\%1.ss1 > nul
copy c:\sparc\medit\mol\%1.cv2 c:\sparc\medit\mol\test.cvt > nul
CONVB
copy c:\sparc\ss\test.ss c:\sparc\ss\%1.ss2 > nul
copy c:\sparc\medit\mol\%1.cv3 c:\sparc\medit\mol\test.cvt > nul
CONVB
copy c:\sparc\ss\test.ss c:\sparc\ss\%1.ss3 > nul
if not exist c:\sparc\medit\mol\%1.cv4 goto finish
copy c:\sparc\medit\mol\%1.cv4 c:\sparc\medit\mol\test.cvt > nul
CONVB
copy c:\sparc\ss\test.ss c:\sparc\ss\%1.ss4 > nul
:finish
ECHO   Putting [%1.ss?], output Smiles String files from CONVB..
C:\NETWORK\FTP\FTP -u staufferst password VTCS1
                                    mput c:\sparc\ss\%1.ss* > nul
CD C:\
ECHO              Deleting all files, [%1.cv?] & [%1.ss?].....
ECHO.
del c:\sparc\medit\mol\%1.cv*
del c:\sparc\ss\%1.ss*
ECHO    Finished running [CONVBALL %1].
ECHO.
ECHO ON
```

Text Frame 70: CONVBALL.BAT

There are several items in Text Frame 70 which warrant elaborations. Since there are four fragments generated for each molecule and each fragment must be named "..\TEST.CVT" for CONVB to run, an intermediate naming system must be used. The system used by CHIRULE and CONVBALL.BAT involves an eight character file name provided by the user, along with a three character extension "CV1, CV2, CV3, CV4", one for each of the fragment MOL files. The CONVBALL.BAT batch file takes the file name as a parameter, and copies the four files "FILENAME.CV1, ..., FILENAME.CV4" from VTCS1 to the local computer

workstation. One at a time, these files are (1) renamed "..\TEST.CVT", (2) copied to the appropriate directory, (3) CONVB is run to generate the SMILES string, and (4) the resulting file "..\TEST.SS" renamed to "FILENAME.SS1, ..., FILENAME.SS4". After this, all of the SMILES string files for the four fragments are copied back to VTCS1, and deleted from the local computer workstation. All of these steps are completed by CONVBALL.BAT.

The final transfer occurs prior to calculation the property values for each of the fragments in SPARC. CHESS generates a file that SPARC's batch mode will accept (BATCH.DAT). This must be transferred to the local computer workstation. SPARC's batch mode then accepts this file, calculates the appropriate values, and writes them out to a results file

```
@echo off
C:
echo.
echo     Obtaining SPARC values for molecule fragments......
echo.
echo         Getting SPARC Batch file BATCH.DAT......
C:\NETWORK\FTP\FTP -u staufferst password VTCS1
                get [.fgroup]batch.dat c:\sparc\batch.dat > nul
echo.
echo     Run SPARC Batch mode with BATCH.DAT and BATCH.OUT.
                                        When finished.
pause
echo         Removing ANSI codes from BATCH.OUT......
c:\wordproc\wp50\wp/D-c:\wordproc/M-sparcall
REM - Running WordPerfect blanks the screen, so re-draw all of
                                        what happened...
echo.
echo.
echo C: TO-SPARC
echo.
echo     Obtaining SPARC values for molecule fragments......
echo.
echo         Getting SPARC Batch file BATCH.DAT......
echo.
echo     Run SPARC Batch mode with BATCH.DAT and BATCH.OUT.
                                        When finished.
echo Pause. Hit any key to continue....
echo.
echo         Removing ANSI codes from BATCH.OUT......
echo.
echo         Putting SPARC Batch file BATCH.OUT......
C:\NETWORK\FTP\FTP -u staufferst password VTCS1
                put c:\sparc\batch.out [.fgroup]batch.out > nul
echo.
echo     Finished running TO-SPARC.BAT
echo.
```

Text Frame 71: TO-SPARC.BAT

(BATCH.OUT). This file is then automatically edited to remove the ANSI escape sequences that SPARC places in the file to highlight various headings. This file is subsequently transferred back to VTCS1. The batch file TO-SPARC.BAT, shown in Text Frame 71, performs the transfers.

The removal of the ANSI escape sequences from SPARCs batch mode output file "BATCH.OUT" is accomplished automatically using a macro in WordPerfect v5.0. One of the command line options while starting up WordPerfect is "/M-SPARCALL" which activates execution of the macro "SPARCALL.WPM" immediately after the program is started. This macro, shown in Text Frame 72, reads the file C:\SPARC\BATCH.OUT in as a text file, then does a global search/delete for {ESC}[1m (the ANSI escape sequence for a bold). The macro then does another global search/delete for 22 (the ANSI code for underline) and replaces it with nothing. It then saves the result as a text file to C:\SPARC\BATCH.OUT, overwriting the existing file, and exits WordPerfect. The resulting ASCII file is able to be read in by CHESS.

```
    File          SPARCALL.WPM
1 - Description   Rid batch mode output of extra junk
2 - Action

  {DISPLAY OFF}
  {Text In/Out}12c:\sparc\batch.out{Enter}
  {Replace}n{^V}{Esc}[1m{Search}{Search}
           {Home}{Home}{Up}
  {Replace}n{Right}{Right}{Del}22{Search}{Search}
           {Home}{Home}{Up}
  {Text In/Out}11c:\sparc\batch.out{Enter}y{Exit}ny
```

Text Frame 72: SPARCALL.WPM

In the CHIRULE development strategy, as in the Prolog strategy, the file transfer window (FTP) could be eliminated by using an NFS link directly to VTCS1 to transfer all files. This would be especially advantageous to CHIRULE's development strategy because of the interaction required with the SMILES string generator that runs on the computer workstation.

Bringing together a number of computer applications running on various computer platforms across an integrated network demonstrates the basic idea behind distributed computing applications. These types of applications can take advantage of the strengths that specialized computers can provide, bringing the capabilities of 3-D graphics engines, database servers, and super computers to the local user's desktop.

VII.D. Loading Rules and Running Systems

This section collects all of the information covered in earlier sections of this chapter and combines it into a single example. The following demonstrations will present the necessary steps in using CHESS to run both OPTIMER and CHIRULE. Instructions for each software package is contained in Section VII.A, while the specifics behind each of the steps can be found in previous chapters (molecular formats in Chapter III.A, KEY in Chapter IV.B, Similarity

Searching in Chapter V.C, and Prolog Rulebases in Chapter VI.D).

VII.D.1. CHESS

CHESS can be loaded by logging onto VTCS1, changing directory to [STAUFFERS.FGROUP], and starting up VPI Prolog. The latter can be done by (1) typing "HCC" and at the Prolog prompt "?-" typing "consult res", or (2) typing only "HCC RES". HCC is a VAX/VMS symbolic definition (HCC :== $user:[roach.deighan]Prolog.exe) that can be found in the users LOGIN.COM. Either of these methods will load the file RES.HC which serves as a master file for CHESS, as well as CHIRULE. This file in turn loads all of the other files that contain the Prolog code for CHESS.

RES.HC can be broken down into four sections: the first loads all of the Prolog code, the second describes each of the KBs, the third contains the predicates that load all of the static KBs and generates all of the dynamic KBs, and the last section contains a number of standard Prolog predicates that are used by all modules. The KB descriptions in section two are provided only internally for documentation purposes. They are not required by Prolog (refer to section VII.B for a complete description). Loading of the KBs required by CHIRULE is described in the following

paragraphs. Appendix F lists the standard Prolog predicates in the module RES.HC.

The first section of RES.HC that loads the Prolog code requires some additional explanation. CHESS is broken down into ten modules, each of which covers a different area of expertise. These modules are loaded one at a time by the predicate "consult fg.hc". Table XII shows the areas addressed by each of the modules. In addition to loading each of the modules, this section of RES.HC defines each of the KBs used by CHESS (and CHIRULE) as dynamic, since each will be expanded as additional molecules are added to the system. Refer to Chapter VI for a further explanation of dynamic predicates.

Once each of the modules has been loaded by RES.HC as described above, CHESS is ready to be used. Two separate examples of how CHESS can be used to develop applications follow: OPTIMER and CHIRULE. (The actual loading of the knowledge bases that have been developed for CHIRULE are discussed in the last section of this Chapter.)

VII.D.2. OPTIMER and KEY

The goal of OPTIMER is to direct users to a location within a Chiral Chromatography Application Guide where the scientists can locate molecules similar to their target molecule. Upon finding a similar molecule, the scientist

Table XII: CHESS Program Modules

RES.HC - Master file, loads all other modules. Predicates for loading static KBs as well as generating dynamic KBs, and a number of common prolog predicates

FG.HC - All functional group predicates

CHEM.HC - Predicates looking for atoms or bonds within the molecular structure

RING.HC - Predicates looking for rings within molecular structures

MOL.HC - Predicates for loading and saving MOL files

SMILES.HC - Predicates for converting molecule KB into format required by CONVB to generate SMILES strings, as well I/O predicates for writing/reading files to/from CONVB.

SPARC.HC - Predicates that handle writing/reading files to/from SPARCs batch mode.

NEW.HC - Testing predicates and other new predicates that are undergoing testing.

KEY.HC - Inference engine predicates to load and run rulebase generated by the automated knowledge extraction utility KEY.

CHIRULE.HC - Predicates for fragmenting molecules, and performing similarity searching of KBs.

can employ its column as a starting point for developing a chiral separation method for their molecule. This process is divided into two tasks: first developing the OPTIMER system (using KEY), and then running the resulting rulebase to answer a user's question. The person involved in the first task is referred to as the developer, whereas the person involved in running the rulebase is called the user. Chapter IV.E contains a more general description of using KEY to generate rulebases.

To run KEY, change to the directory containing KEY6.EXE and type "KEY6" to start the acquisition of the grid as described in Chapter IV.C. Any element, trait, or grid files that are saved, or any such files that are needed, must be located in the same directory as the main program file. Proceed with the analysis of the grid and the rulebase generation as described in Chapter IV.D and IV.E. The users last decision addresses which system (Turbo-Prolog or CHESS) the rulebase is intended for. The file "T-Prolog.SAV" must reside in the same directory as KEY if a Turbo-Prolog rulebase is to be generated.

If Turbo-Prolog is to be used to run the resulting rulebase, the rulebase file must be copied to the Turbo-Prolog directory. To run the rulebase, start up Turbo-Prolog, read in the rulebase file, and execute it. After inputting answers regarding which trait pole most accurately

reflects the user's molecule (see Text Frame 27), a list of suggestions of where to look within the application guide for similar molecules will be generated (see Text Frame 28).

If CHESS is to be used to run the resulting rulebase, the rulebase must be transferred to VTCS1. The user must obtain a MOL file of the molecule or molecules of interest. This can be accomplished using MACCS (see Section VII.A.3). After transferring the MOL files to VTCS1, start up CHESS as described in the previous section. Note that Table XII contains the Prolog module KEY.HC in addition to the other Prolog modules. This module contains the inference engine that will not only execute the rules, as does the Turbo-Prolog inference engine, but also take advantage of the general chemical knowledge hooks built into the CHESS shell (see Chapter III, Table III, and Text Frame 26).

To begin execution of the rulebase, type "run" and enter the name of the rulebase as shown in Text Frame 48. The user will then be prompted for the name of a molecule for which a chiral separation method is to be developed (see Text Frame 25). The input name can represent one of two things: the name of a molecule in the Molecule KB developed for CHIRULE (described in the following section) or the name of a MOL file that resides in the current directory (including the extension).

This will start the execution of the OPTIMER rulebase, taking advantage of the "hooks" built into CHESS (see Text Frame 26). After the inference engine has executed all of the rules, it will print out the results and a list of the responses for each of the traits (see Text Frame 49). Here, as in the Turbo-Prolog system, the results consist of page numbers within the application guide pointing to locate molecules similar to the user's target molecule (see Text Frame 28).

There is one predicate in inference engine developed to run the code generated by KEY that needs to be discussed. Predicate "check4mol" must be modified for each rulebase that will be run by KEY. In this case, the KEY rulebase will be looking at molecule names. "check4mol" determines if the molecule input by the user already exists in the Molecule KB, or if it needs to be read in from a MOL file. If other rulebases do not deal with molecules, "check4mol" will have to be modified accordingly. For more information, consult Module "KEY.HC" in Appendix F.

VII.D.3. CHIRULE

Using CHIRULE requires that the development strategy described in section VII.C.3 be implemented (see Figure 16). Refer back to that section for the details of how the transfers take place. Figure 17 displays a timeline diagram

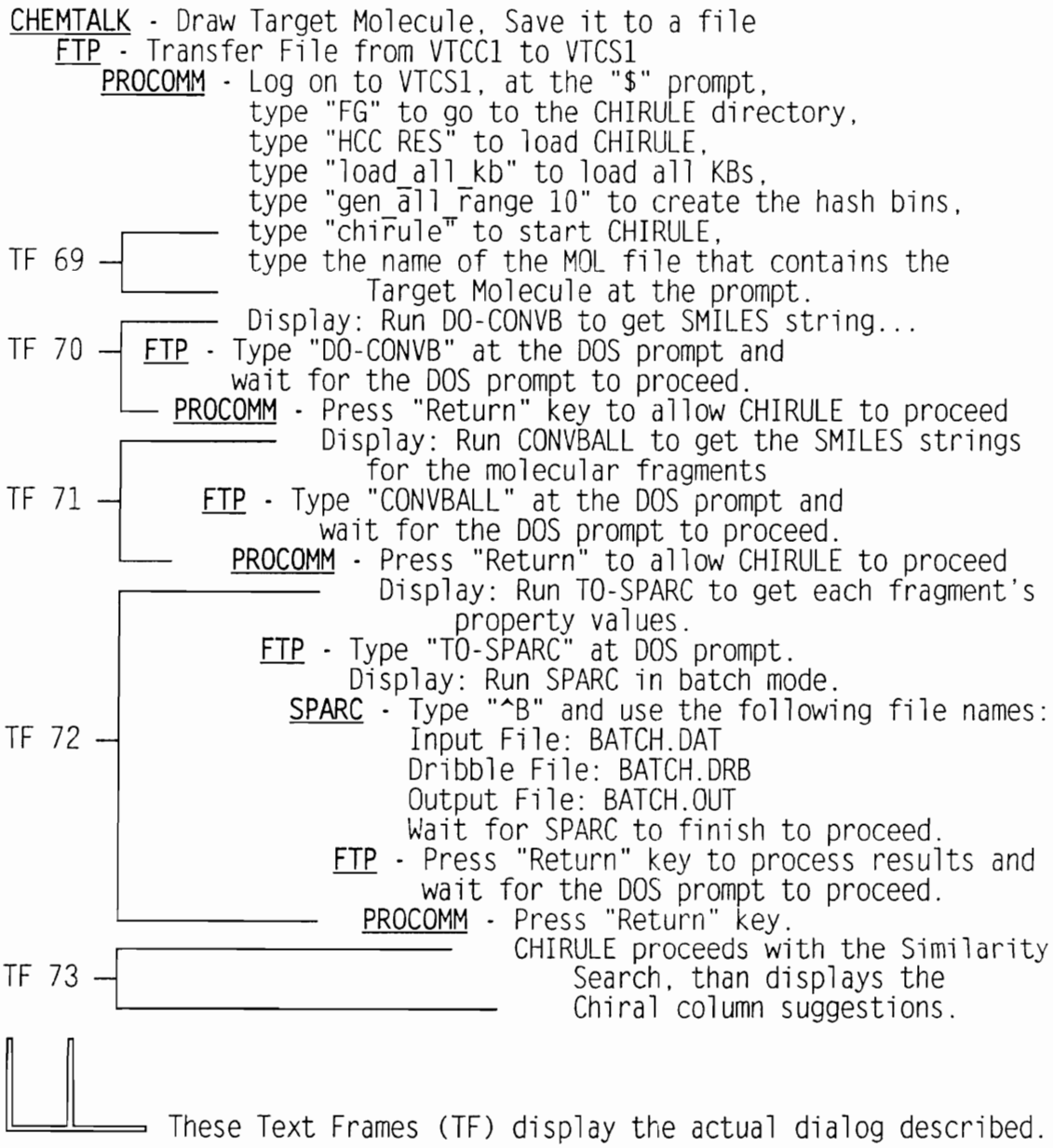


Figure 17: Using CHIRULE's Development Environment

of using CHIRULE. Note the references to the Text Frames that depict the actual dialog for the steps indicated.

Running CHIRULE begins in the ChemTalk window using MACCS to draw the target molecule and generate its MOL file. The MOL file is then transferred to VTCS1 using the FTP window as indicated in Figure 17. CHESS must be loaded by switching to the Procomm window. Follow the instructions in section VII.D.1 for loading CHESS.

In addition to loading CHESS, all of the KBs must be loaded before CHIRULE can be run. The predicate "load_all_kb" will load all of the static KBs (Molecule, Fragment, Fragment Value, and Chiral Column KBs). The predicate "gen_all_range <delta>" will generate the runtime Fragment Value Range KB necessary for the similarity searching. Execute each of these predicates from the Procomm window by typing them at the VPI Prolog prompt after CHESS has been loaded. A delta value of 10 has proven to be most useful. Refer to Chapter VIII for a further discussion on delta values.

Once all of the KBs have been loaded, CHIRULE can be run. At the VPI Prolog prompt "?-" in the Procomm window, type "chirule" (in lower case letters) to begin. Then input the file name representing the target molecules MOL file (including extension) at the prompt as shown in Text Frame 73. CHIRULE then uses CHESS to (1) read in the

molecule, (2) display the name of the molecule, (3) add it to the runtime Molecule KB, and (4) make sure the molecule has a chiral carbon atom.

```
?-
?- chirule
      ***** Welcome to CHIRULE!! *****

Enter the file spec (including the extension) for the Chiral Molecule
     you wish to determine a Chiral column: -> diactxbt.mol
       Molecule name: {1.3-DIACETOXY-BNTONC}
```

Text Frame 73: MOL file name for CHIRULE

The next three tasks illustrate aspects of network integration made possible by the development strategy already presented (see Section VII.C.3 above). The tasks demonstrate (1) acquiring the SMILES string for both the target molecule and its fragments, and (2) calculating property values for each of the fragments. They show how CHIRULE, running on a VAX, can take advantage of programs that are written for the PC workstation. This is done by transferring files between the two computers.

The first task involves generating the SMILES representation for the target molecule. CHIRULE (1) writes the modified MOL file TEST.CVT that the program CONVB will accept and (2) prints a message in the PROCOMM window indicating the batch file DO-CONVB.BAT should be run. This batch file will run CONVB, which generates the SMILES string for the target molecule. Switch to the FTP window (by tapping the DESQview key [the Alt key] rapidly twice in succession) and type "DO-CONVB" at the DOS prompt

(DO-CONVB.BAT can be found in Text Frame 69). When the batch file finishes running, switch back to the Procomm window and press "RETURN" to proceed. The SMILES string for the target molecule is read into CHIRULE and added to the Molecule KB. Text Frame 74 contains the dialog from both windows.

```
                    PROCOMM WINDOW DIALOG...

Run DO-CONVB.BAT now to get the SMILES string for the molecule
          1.3-DIACETOXY-BNTONC.  Then
          hit RETURN when ready to proceed.....

                    FTP WINDOW DIALOG...

C>DO-CONVB

     Getting TEST.CVT from VTCS1.....

     Running CONVB to generate the Smiles String....

     Putting TEST.SS to VTCS1.....

C>
```

Text Frame 74: Generating SMILES String for CHIRULE

The next task for CHIRULE is to fragment the target molecule around the chiral atom and obtain the SMILES strings for the four individual fragments (see Text Frame 75). After fragmenting the molecule, CHIRULE prints the atom number (from the MOL file) of the chiral carbon, then asks for the file name to use for the MOL files for each fragment. After inputting this name, printing the atom numbers in each fragment, and writing the four MOL files, the user is instructed by CHIRULE to run CONVBALL.BAT to obtain the SMILEs strings for each of the fragments. Switching from the Procomm window to the FTP window, the user types "CONVBALL filename" at the DOS prompt (CONVBALL.BAT can be found in Text Frame 70). When the

batch file is finished running, switch back to the Procomm window and press RETURN for CHIRULE to continue. after making sure the fragment SMILES strings do not already exist, they are read in, displayed for the user, and entered in the Molecular Fragment KB. If the fragment SMILES strings already exist, they are discarded since there is no need to create duplicate entries.

```
                    PROCOMM WINDOW DIALOG...

Chiral atom number 2 found in 1.3-DIACETOXY-BNTONC.  Please enter
     a file name for saving Molecule fragment files (NO EXT): -> DAB
  The fragment list for 1.3-DIACETOXY-BNTONC is
     [((2 3 4 5 6 7 8) (2 9 10 11 12) (2 1) ())]

Run CONVBALL.BAT now with the fragment file [DAB] to get the SMILES
     string for the fragments of 1.3-DIACETOXY-BNTONC.  Hit
     return when ready to proceed [i.e. CONVBALL DAB].....

  The smiles string list is
       [O=C(OCC)C O=C(O)C C ]
  Making sure the information doesn't exist in the runtime
        fragment database...

                    FTP WINDOW DIALOG...

C>CONVBALL DAB
     Obtaining Smiles strings for fragments of DAB.....

        Getting [DAB.cv?], input files for CONVB......
             Running CONVB......

        Putting [DAB.ss?], output Smiles String files from CONVB....
             Deleting all files. [DAB.cv?] & [DAB.ss?].....

     Finished running [CONVBALL DAB].
C>
```

Text Frame 75: Generating Fragment SMILES Strings for CHIRULE

The last task illustrating the integration required for CHIRULE involves obtaining the property values for each of the four fragments. This is shown in Text Frame 76. If it is found that values already exist for all of the fragments, CHIRULE will proceed to the similarity search. If they do not all exist, the name of the molecule, its SMILES string, and each of the fragment's SMILES strings are printed as SPARC's batch mode input file is generated. Then the user

is instructed to run TO-SPARC.BAT, shown in Text Frame 71. After switching to the FTP window and typing TO-SPARC at the DOS prompt, the batch file copies SPARC's batch mode input file to the local computer workstation, and then pauses and instructs the user to run SPARC's Batch Mode. After switching to the SPARC window, typing "ALT-B" brings up the batch mode reader. The user is prompted for three files: the input file, the dribble file, and the output file. The default input file, BATCH.DAT, must be used. The dribble file is optional. The output file should be given the name BATCH.OUT. After SPARC finishes processing the batch file, switch back to the FTP window and press RETURN to continue with TO-SPARC.BAT. WordPerfect is called to remove all of the ANSI escape sequences (see Text Frame 72) before

```
                    PROCOMM WINDOW DIALOG....

  Checking to see if the SPARC values already exist...

Getting values for the molecule 1.3-DIACETOXY-BNTONC
     with the Smiles string [O=C(OCCC(OC(=O)C)C)C].....
     Smiles string {O=C(OCC)C} being used...
     Smiles string {O=C(O)C} being used...
     Smiles string {C} being used...
     Last fragment is a Hydrogen, using the Smiles String {}.

Run TO-SPARC.BAT now, followed by SPARC's Batch mode.  Hit return
     when ready to proceed [i.e. TO-SPARC].....

  Saving the molecule data for 1.3-DIACETOXY-BNTONC
     to the SPARC property fragment database...

                    FTP WINDOW DIALOG....

C> TO-SPARC
   Obtaining SPARC values for molecule fragments......

      Getting SPARC Batch file BATCH.DAT......

   Run SPARC Batch mode with BATCH.DAT and BATCH.OUT.  When finished,
Pause, Hit any key to continue....

      Removing ANSI codes from BATCH.OUT......
      Putting SPARC Batch file BATCH.OUT......

   Finished running TO-SPARC.BAT
C>
```

Text Frame 76: Generating SPARC Fragment Values for CHIRULE

transferring the output file \back to VTCS1 and exiting to DOS. The user then switches from the FTP window to the Procomm window where, after pressing RETURN to continue, CHIRULE inputs the file and saves all of the molecular fragment data.

Now that all of the necessary data has been gathered, the similarity search can be performed. No additional user input is required to execute the searching process. The searching algorithms (as described in Chapter V.C) are performed automatically and an ordered suggestion list is displayed for the user. The suggestion list (shown in Text Frame 77) first displays the name of the target molecule, then lists the following information for each column CHIRULE suggests: the column name, the total difference measure, the name of the molecule in the KB that was found to be similar, and a location reference (for the method).

```
= Ordered suggestions [column] for 1.3-DIACETOXY-BNTONC....

+ Chiralcel OB [569.9], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [620.5], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [801.3], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chrownpak CR [818.7], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralcel OB [892.8], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [892.8], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Crownpak CR [910.2], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OB [1047.2], matching ALPHA METHYL PHENETHYL BENZOATE
                                                {Daicel p45}.
```

Text Frame 77: CHIRULE Column Suggestions

After printing the suggestion list, the user is given the opportunity to add the target molecule to the static versions of the knowledge bases. This is particularly

useful in adding new, known separations to the knowledge bases (refer to Chapter VIII for a further discussion). If this is requested, the molecule is added, and the user is prompted for the type of column used and a reference for the separation. Multiple columns for the same molecule are also accommodated.

```
Do you want to add the molecule 1.3-DIACETOXY-BNTONC
    to the static MOLECULE, SMILES, and COLUMN knowledge base?
    [Y=yes, N=no (default)]  Y/N -> y

Saving 1.3-DIACETOXY-BNTONC to the Molecule Knowledge base...
    Saved 1.3-DIACETOXY-BNTONC to the Molecule Knowledge base...
  Saving 1.3-DIACETOXY-BNTONC to Fragment knowledge base...
    Saved 1.3-DIACETOXY-BNTONC to Fragment knowledge base...

Saving 1.3-DIACETOXY-BNTONC to the Column knowledge base...
    What is the Chiral Column used? -> Chiralcel OB
    What is the reference for this Column? -> Daicel p45
    Saved 1.3-DIACETOXY-BNTONC to the Column knowledge base...

    Additional columns to add for 1.3-DIACETOXY-BNTONC?
       (Y/N, default N) --> n
```

Text Frame 78: CHIRULE Adding Molecule to Knowledge Base

These steps have all described running only a few of the predicates designed for the CHIRULE system. Others, such as testing predicates, are covered in detail in Chapter VIII and can be found in Appendix F. CHIRULE and OPTIMER are examples that were constructed with CHESS. This chapter is intended to demonstrate how a developer could take advantage of this shell to construct another expert system.

VIII. Results and Discussion

The objectives of the current research are three-fold: (1) to construct an expert system shell capable of reasoning in the chemical arena, (2) to develop a column suggestion methodology for chiral chromatography, and (3) utilize network integration to add appropriate tools to the expert system shell running on appropriate platforms. Expert systems are of little value in areas where fundamental models and theoretical equations exist, or in fields in which little empirical knowledge is available. Chromatography is a well bounded problem area which is an ideal test bed, and the rapidly emerging need for guides to selection of chiral chromatographic columns for separation of optical isomers is most appropriate.

The utility of CHESS, the Chemistry-Based Expert System Shell, can be seen by examining the success of the chemical systems developed using its capabilities. OPTIMER was developed in a test situation that allowed a non-computer oriented chromatographer to use the personal construct tools of CHESS. These tools permitted the programming novice to develop a rule-based expert system via a simple question/answer dialogue with the software. While CHESS can be used to handle the more straightforward emulation of the decision tree knowledge found in many chiral column

selection guides, it will also handle more complex, evolving empirical problems. CHIRULE, also based on CHESS, is extremely helpful in choosing what column with which to begin development of a chiral chromatographic separation using molecular descriptors.

Both systems rely on molecular representations of molecules, and the capability of reasoning on both functional groups and molecular properties. These capabilities, provided as part of the general chemical knowledge within CHESS, are not found within typical expert system shells. CHIRULE, with its success rate approaching 90% (see Table XIII), and OPTIMER both demonstrate the utility of introducing general chemical knowledge into an expert system.

The previous chapters have outlined in some detail the history, philosophies, and operation of the CHESS, KEY, OPTIMER, and CHIRULE systems. This chapter addresses the testing of the working systems. Following the example set in previous units, the software predicate details follow a discourse section. The former will be of interest to software engineers. The following will demonstrate that the goal of the research mission has indeed been met.

VIII.A. CHIRULE's Results

CHIRULE will always suggest a column to use in developing a method to resolve an enantiomeric pair. The search for a Sim-List molecule (and its column) similar to the target molecule will always succeed. After discussing whether the methodology is reasonable, the questions to be addressed are two fold: (1) does the total difference measure calculated for the column suggested by the CHIRULE system indicate a close match between the target and Sim-List molecules, and (2) is the column suggested a reasonable suggestion.

VIII.A.1. Discussion of CHIRULE's Results

Exhaustive tests of the column suggesting capabilities of CHIRULE were performed. Each molecule found in CHIRULE's Molecule KB was posed as a target molecule. Chapter V describes how the similarity search specifically rules out the possibility of a direct match if the molecule serving as a target molecule is also found within the Molecule KB. The column suggestions generated for the target molecules were compared to known separation methods. For CHIRULE, smaller numbers are better since they represent the weighted, summed property differences between the target and Sim-List molecules. Table XIII summarizes the results of these tests. Consult Figure 18 through Figure 30 for graphical

Table XIII: CHIRULE Testing Results

Chiral Column	Number of Molecules	First Suggestion Supported	First or Second Supported
CrownPak CR Wainer Type III, crown ether	29	90%	90%
ChiralCel OB Wainer Type II, cellulose ester	22	77%	95%
ChiralCel OD Wainer Type II, cellulose carbamate	22	64%	73%
ChiralPak WH Wainer Type IV, ligand exchange	23	83%	87%
TOTALS:	96	79%	88%

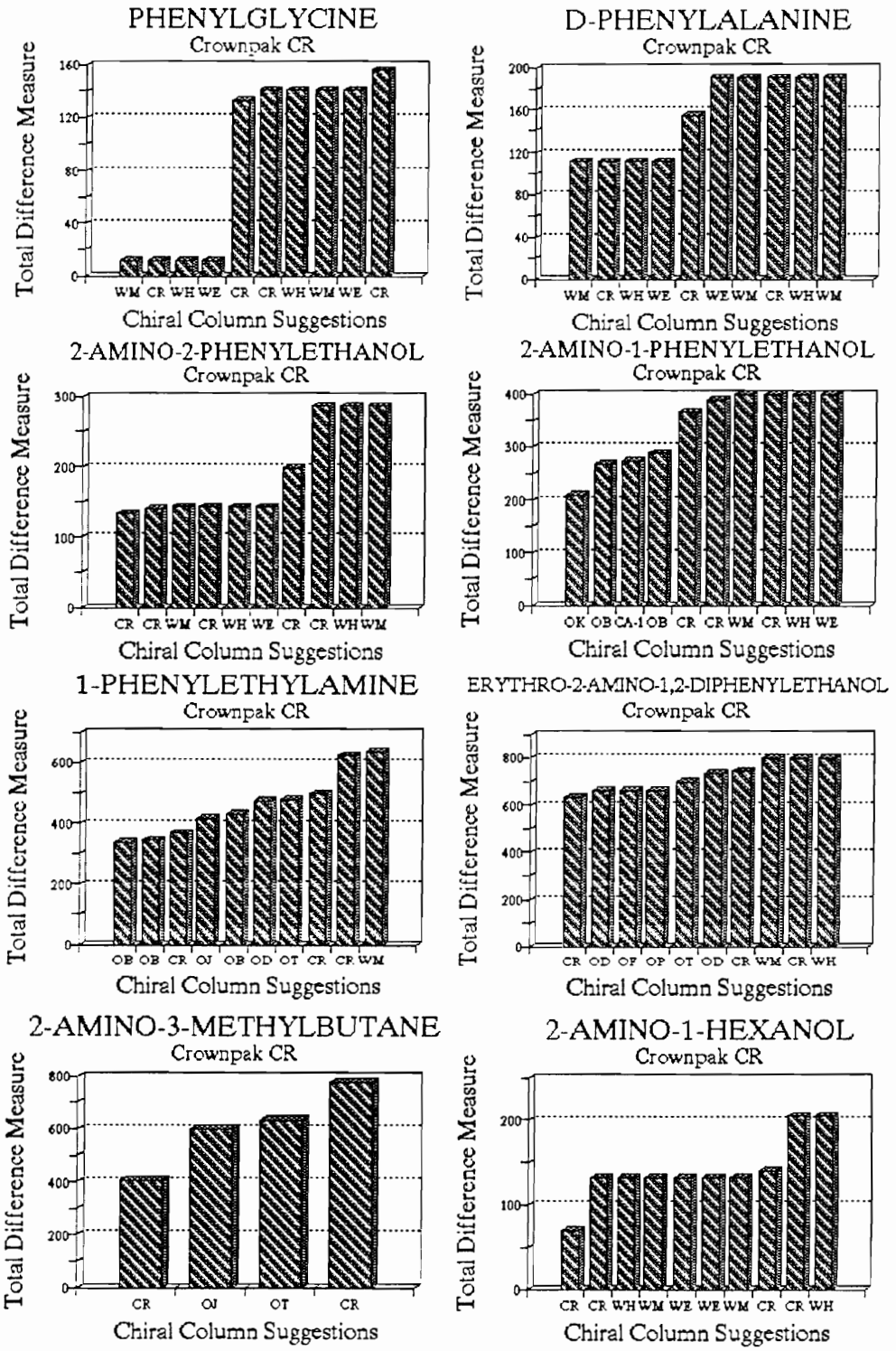


Figure 18: CHIRULE Results for CrownPak CR, Group 1

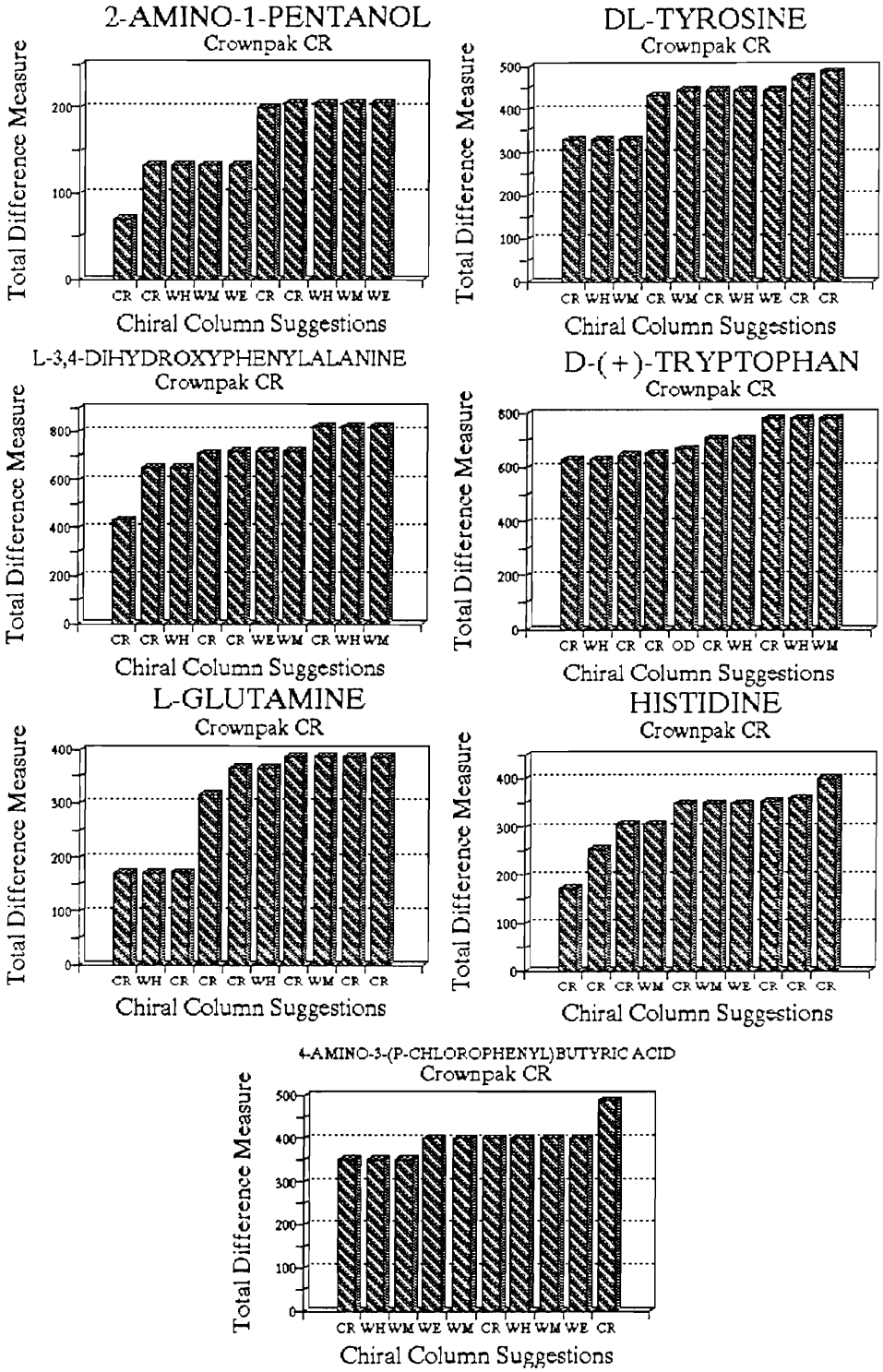


Figure 19: CHIRULE Results for CrownPak CR, Group 2

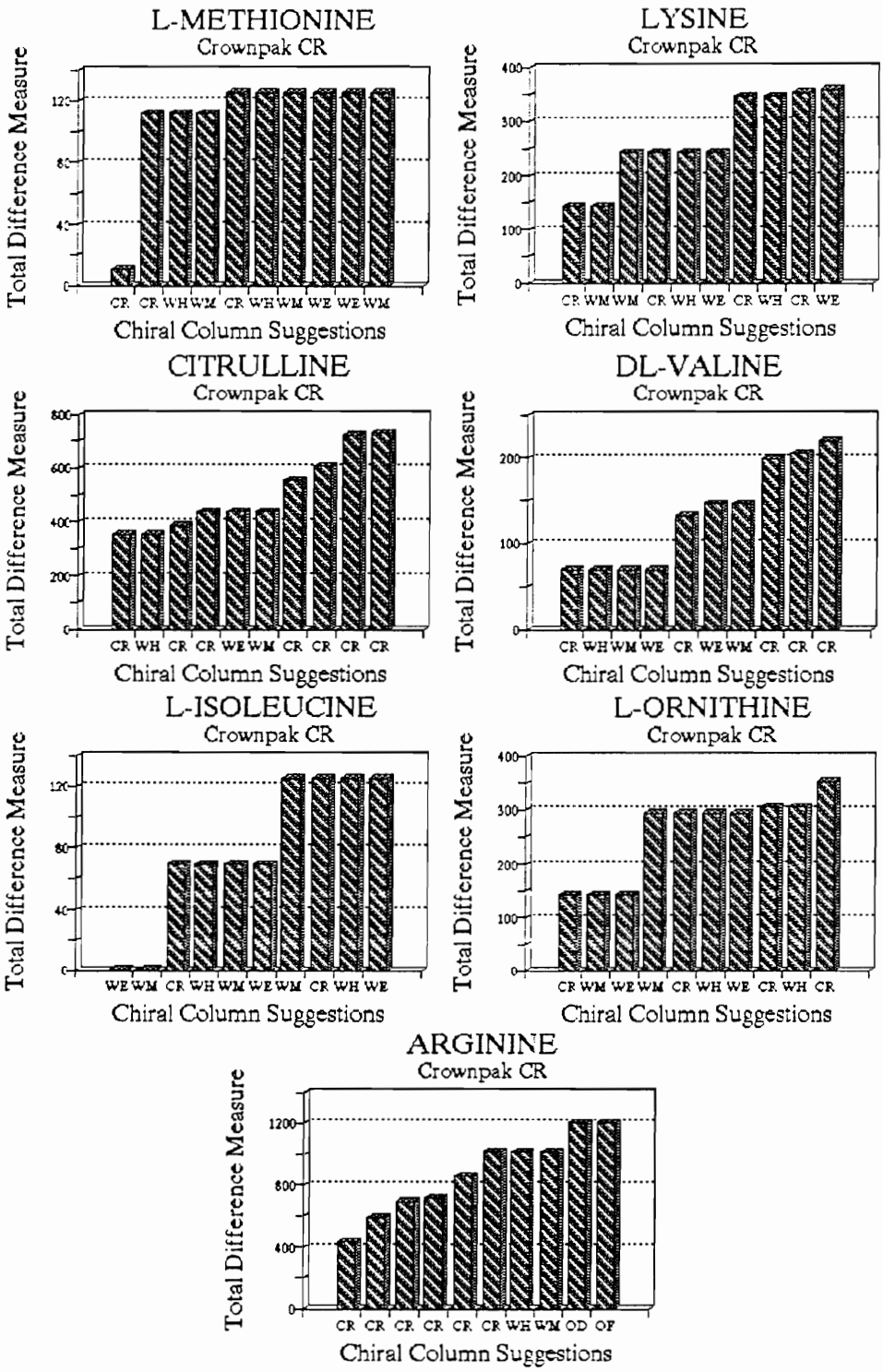


Figure 20: CHIRULE Results for CrownPak CR, Group 3

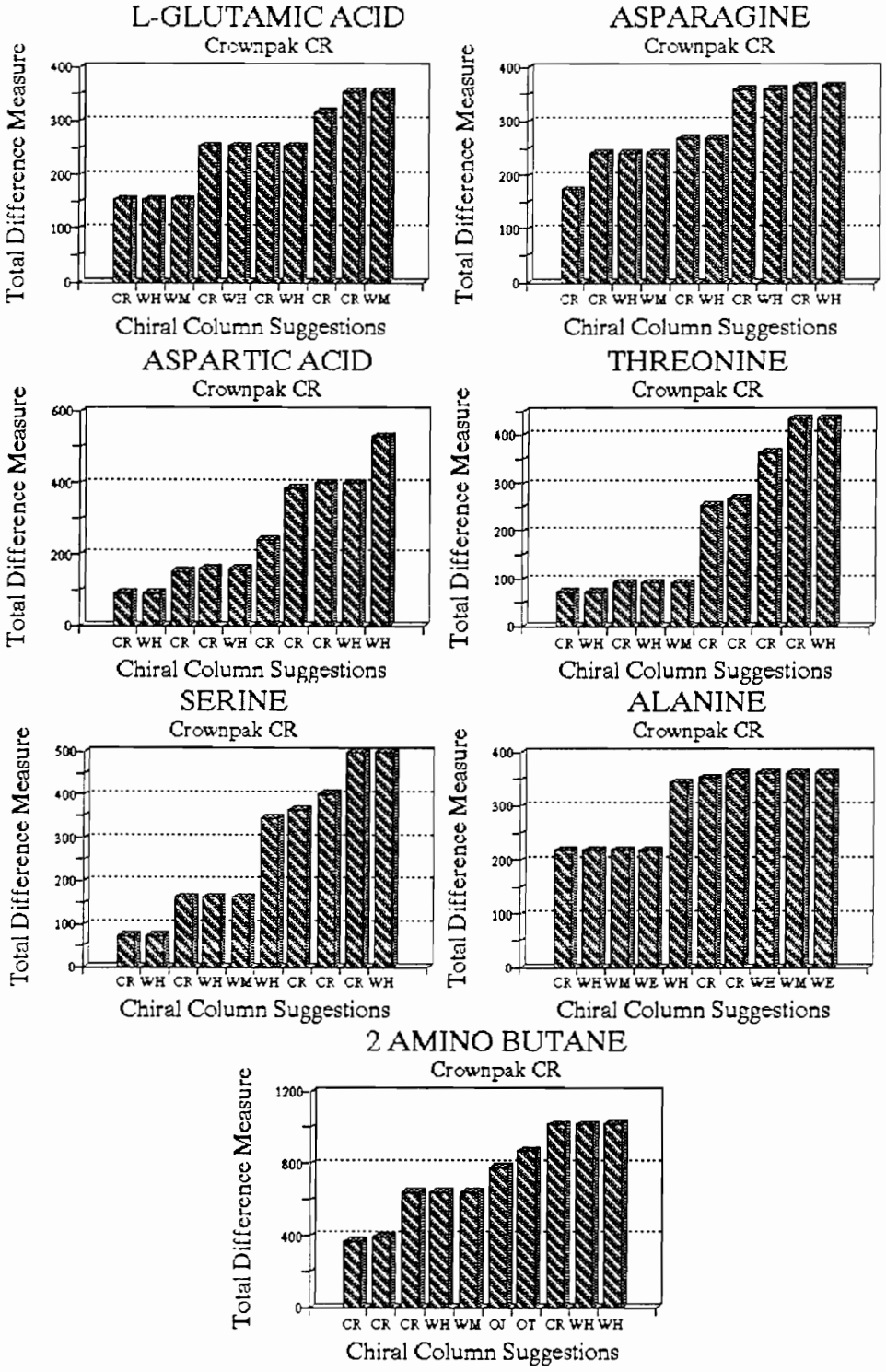


Figure 21: CHIRULE Results for CrownPak CR, Group 4

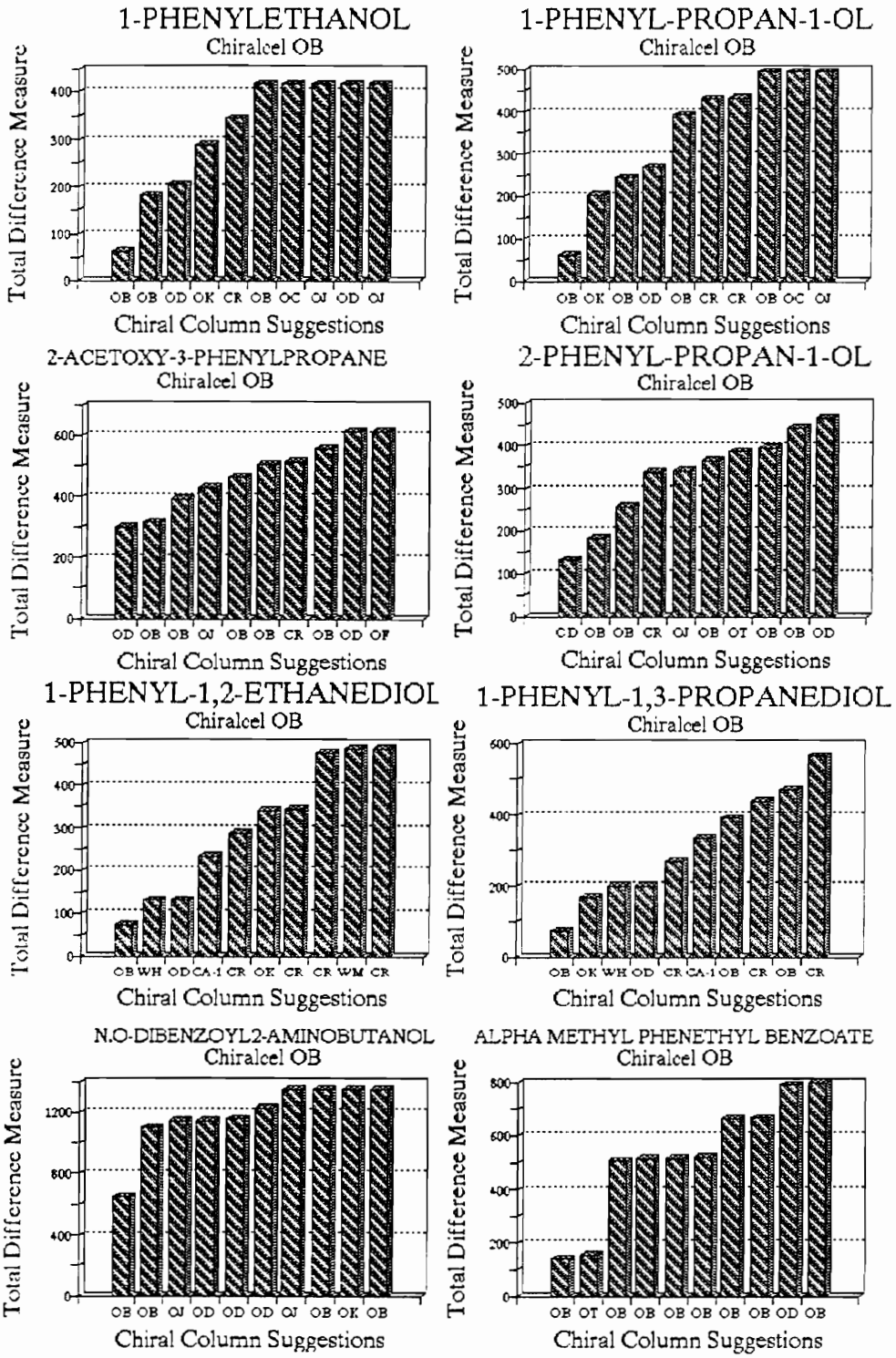


Figure 22: CHIRULE Results for ChiralCel OB, Group 1

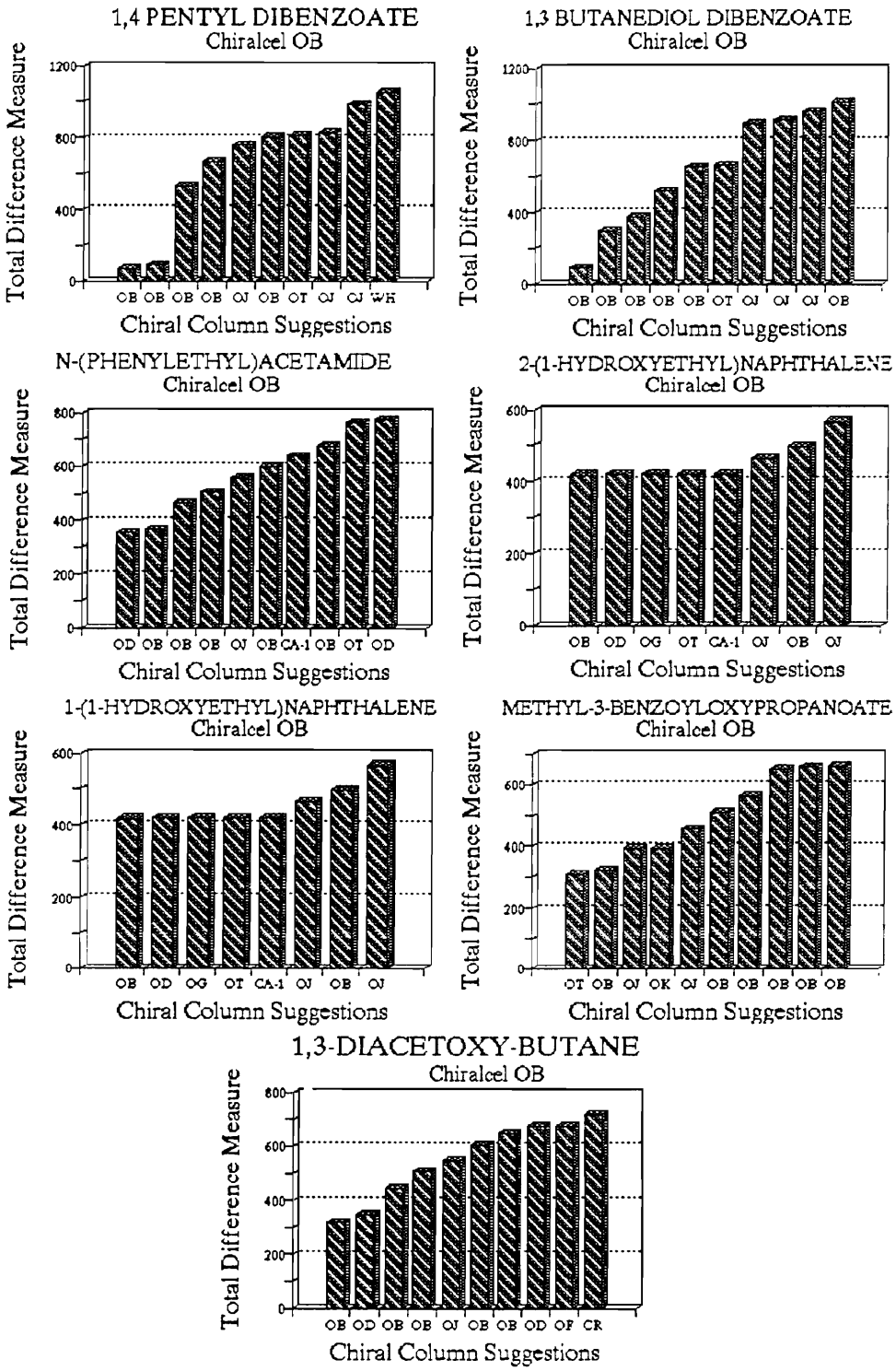


Figure 23: CHIRULE Results for ChiralCel OB, Group 2

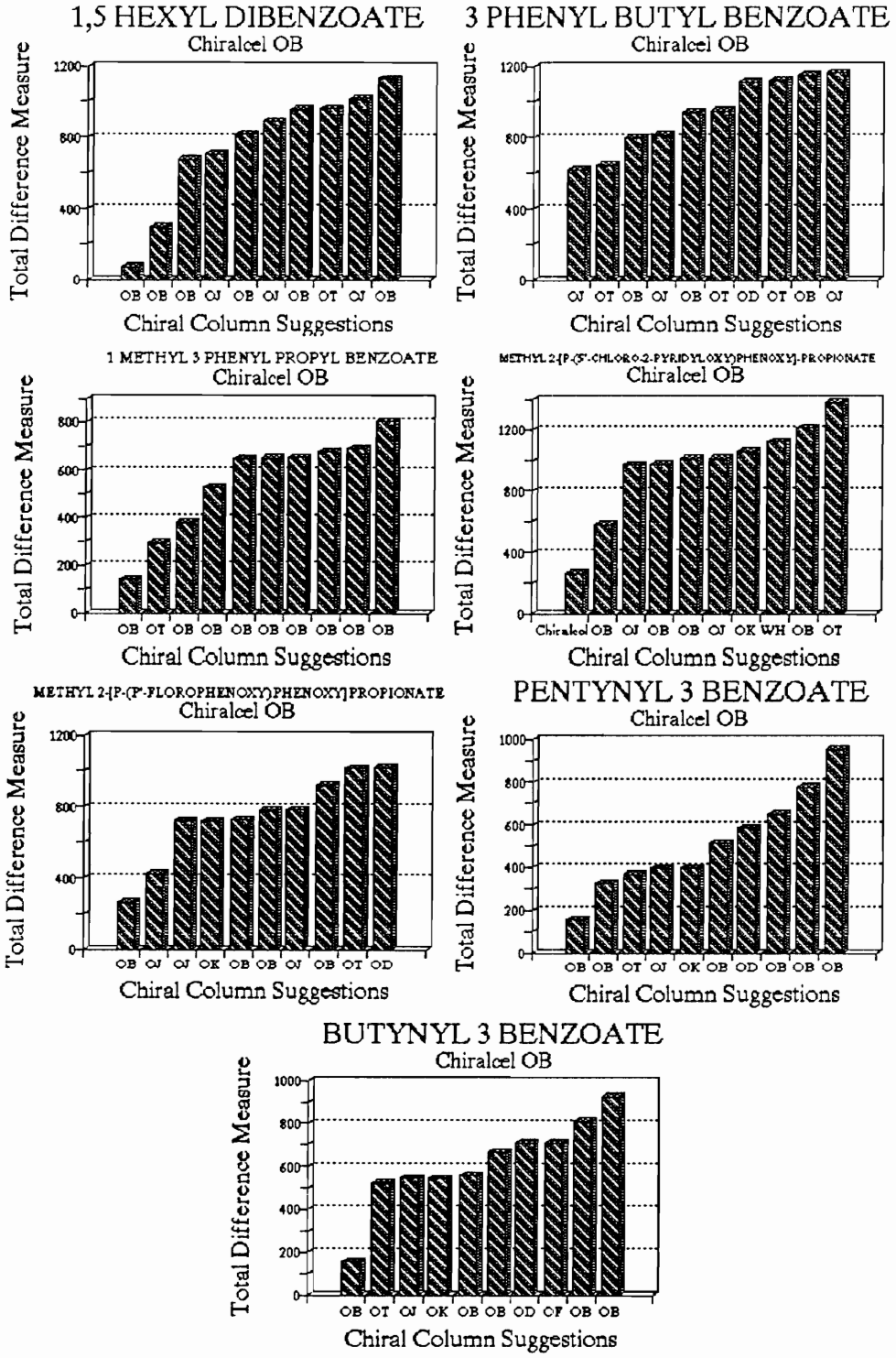


Figure 24: CHIRULE Results for ChiralCel OB, Group 3

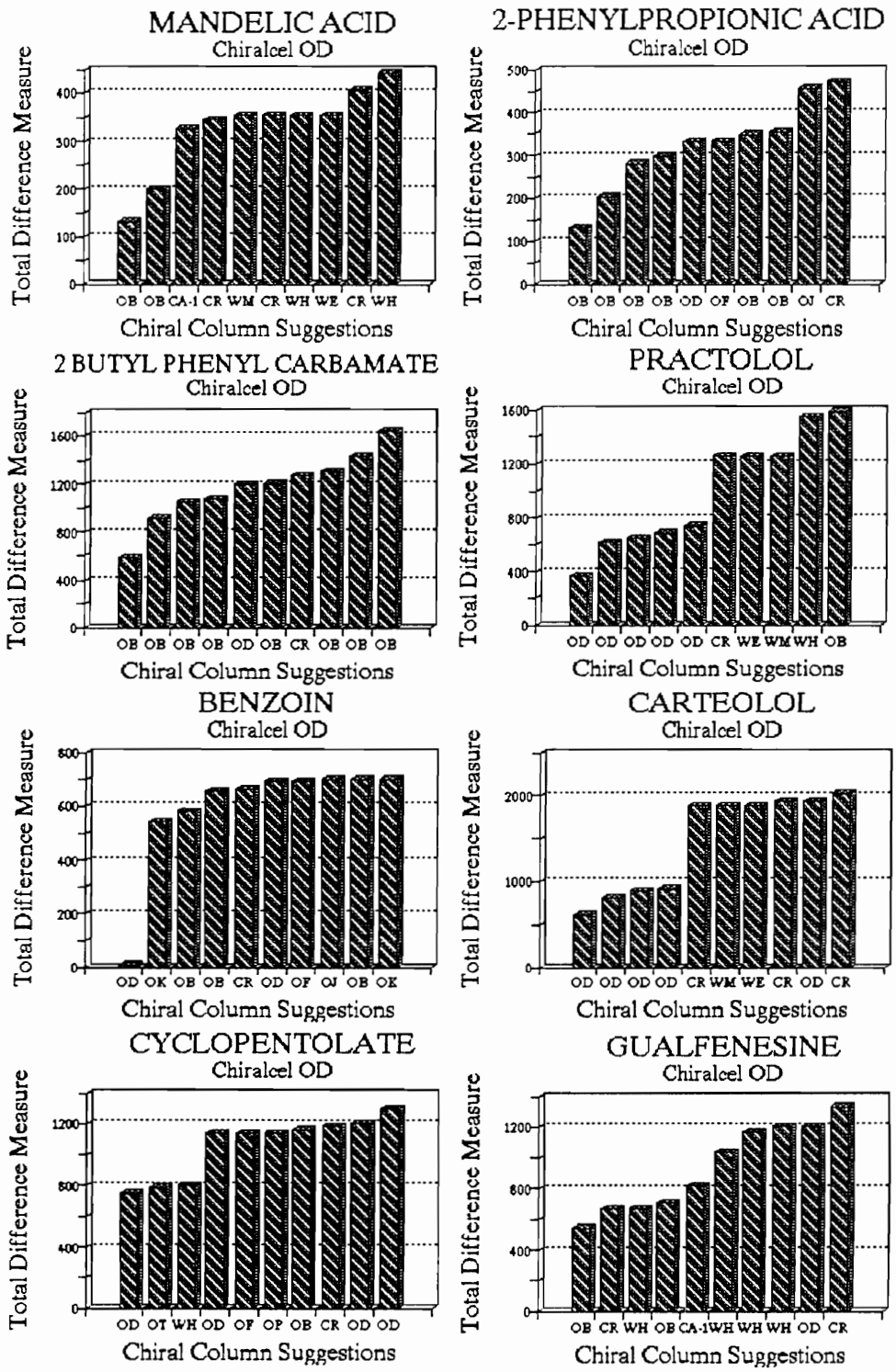


Figure 25: CHIRULE Results for ChiralCel OD, Group 1

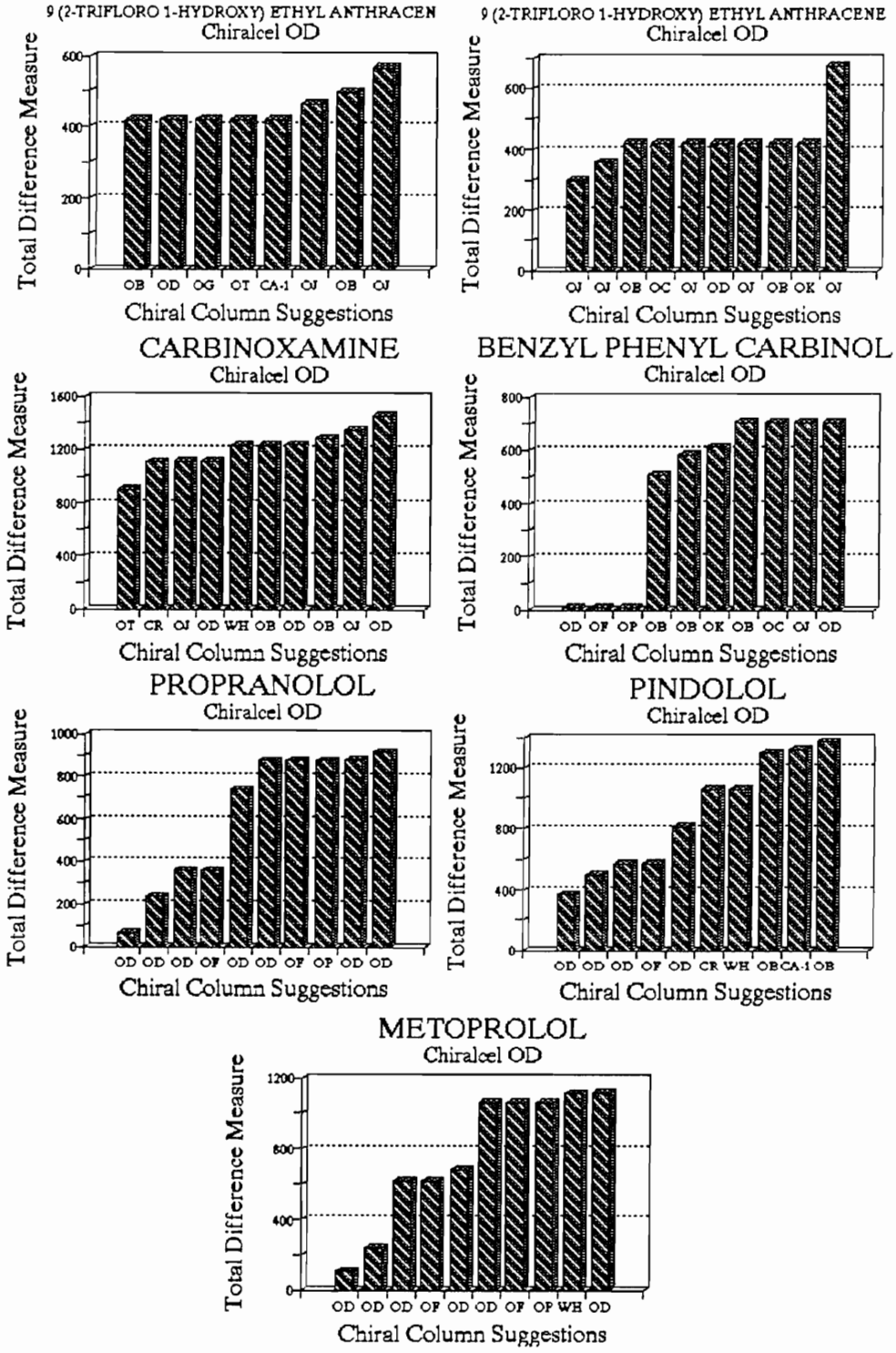


Figure 26: CHIRULE Results for ChiralCel OD, Group 2

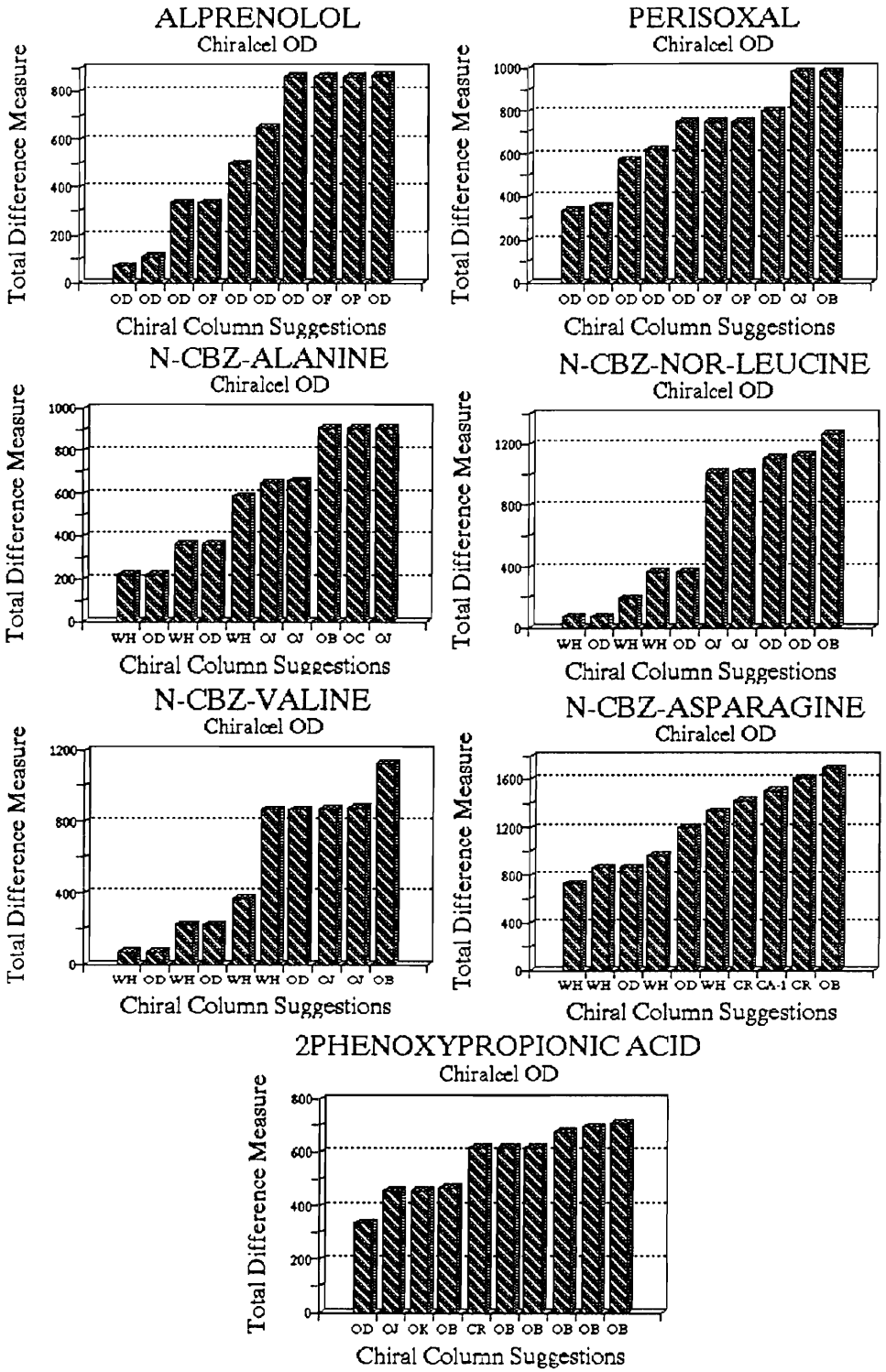


Figure 27: CHIRULE Results for ChiralCel OD, Group 3

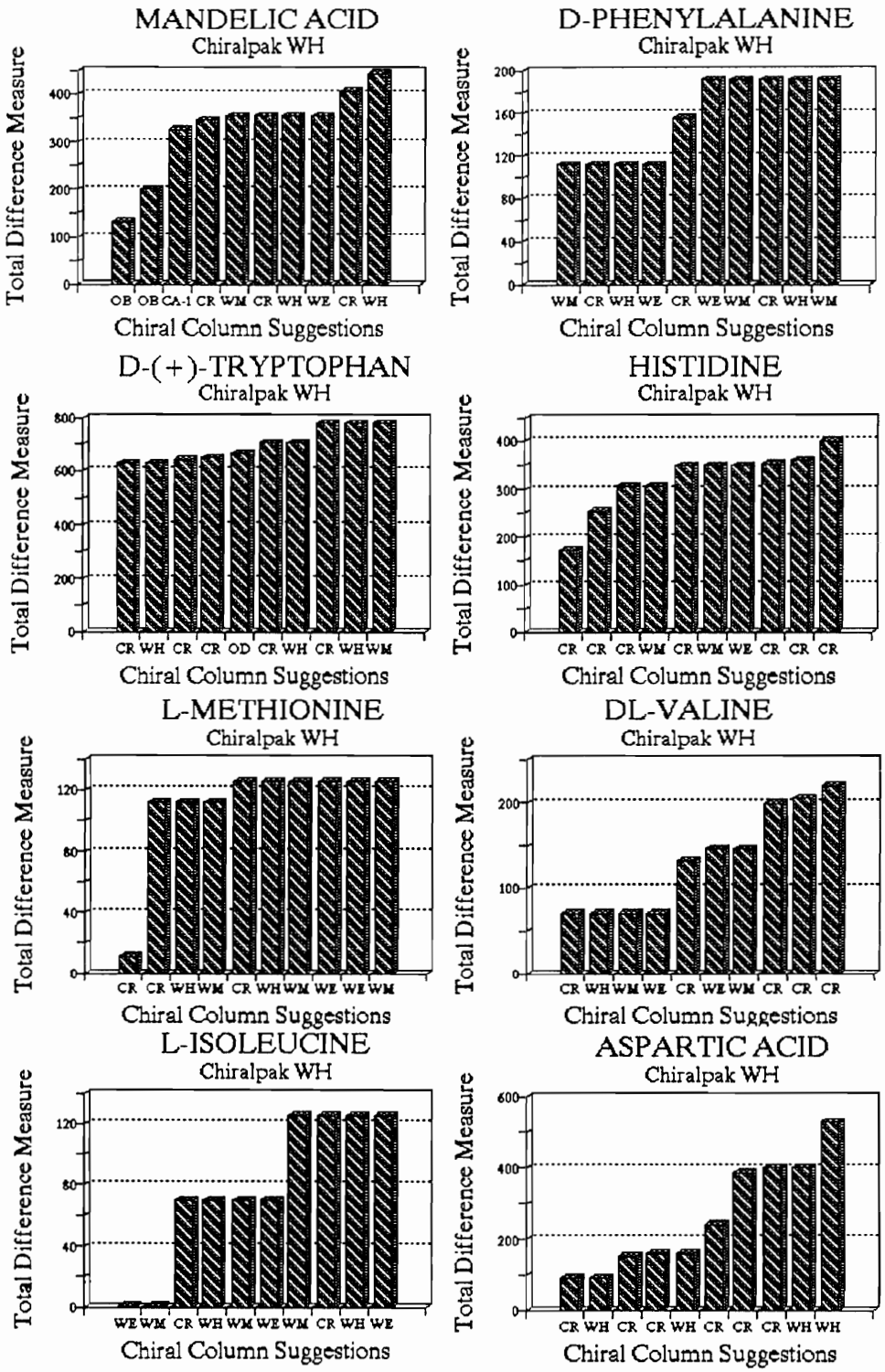


Figure 28: CHIRULE Results for ChiralPak WH, Group 1

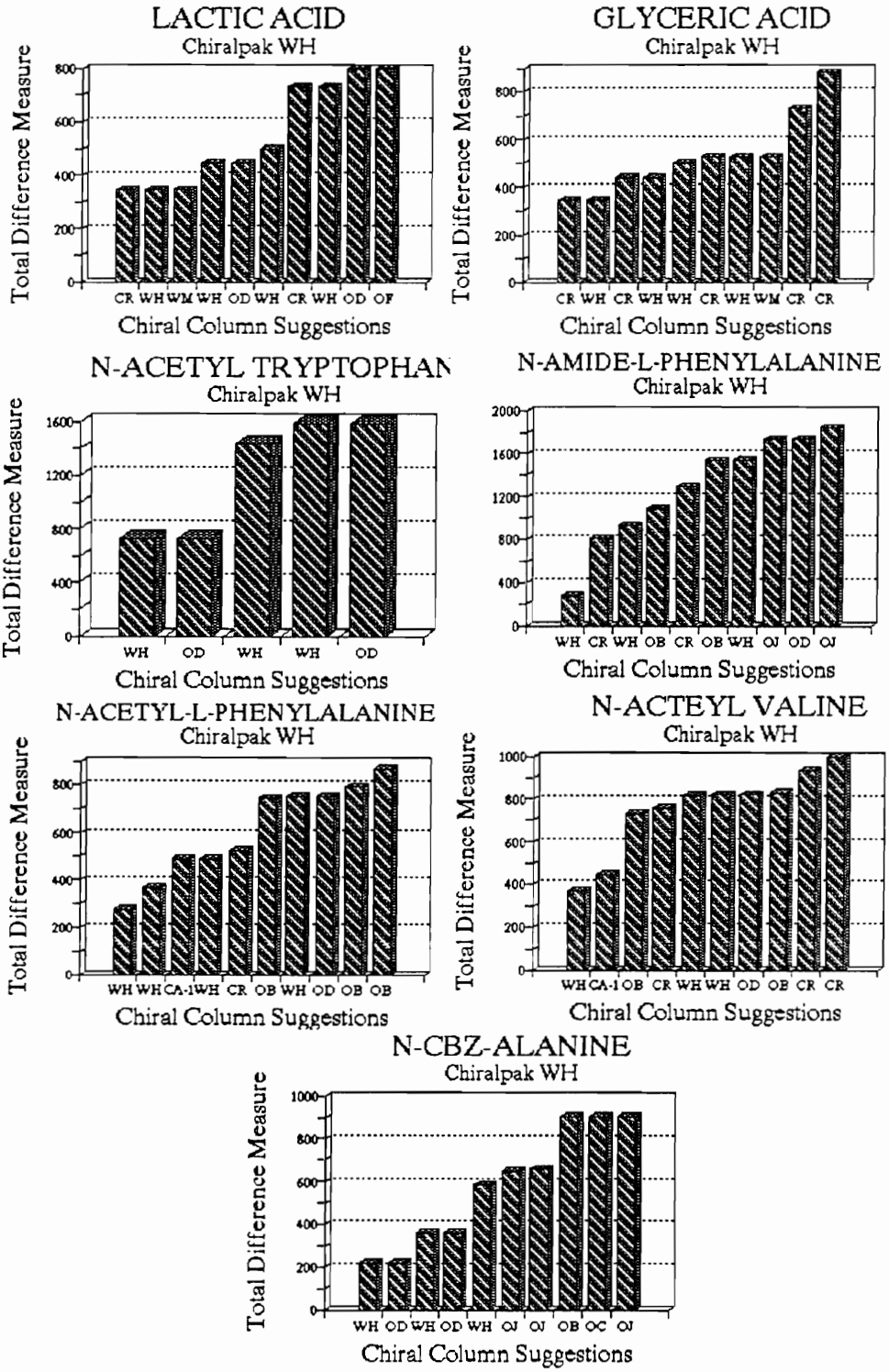


Figure 29: CHIRULE Results for ChiralPak WH, Group 2

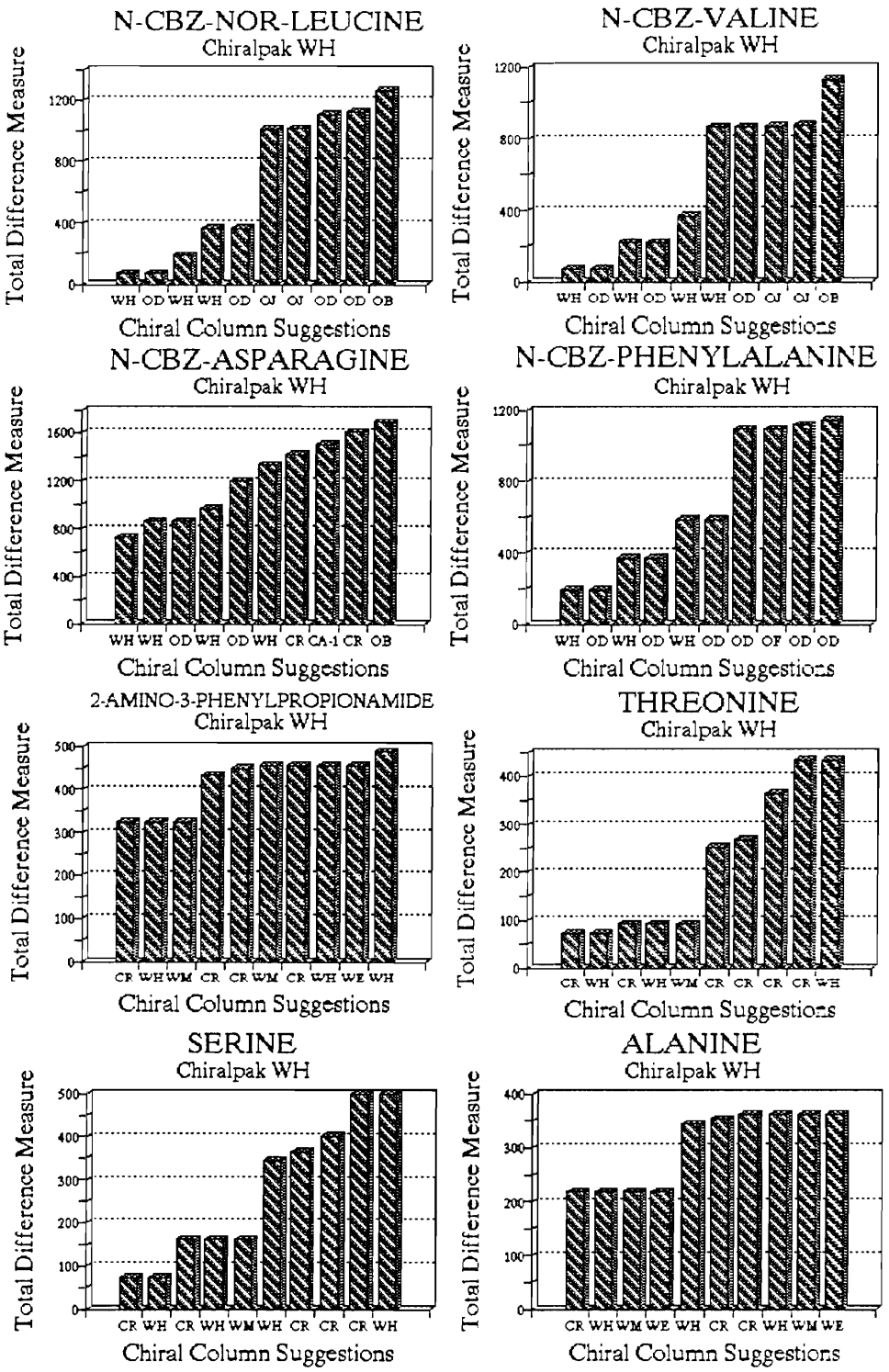


Figure 30: CHIRULE Results for ChiralPak WH, Group 3

representations of the data resulting from these tests. The entry below the molecule name is the reported chiral column. The reader is cautioned that the vertical scales for these figures are altered. Any attempt to do otherwise resulted in either 1) having to change the size of the graphs which precluded their compact presentation, or 2) making it impossible to see many of the data points.

It is not appropriate to judge the chiral column suggestions given by CHIRULE as either "correct" or "incorrect". Rather it seems more relevant to use either the phrase "supported by CHIRULE's KBs" or "unsupported by CHIRULE's KBs". This approach is taken because unsupported column suggestions may in fact be "correct" but simply absent from the Chiral Separations Guide that was used to generate CHIRULE's KBs.

In addition, the scale of the total difference measure used in reporting the results is arbitrary. However, tests suggest that numbers in excess of 2000 are inappropriate, as suggested columns have unsuitable separation features. Numbers below 100 involve columns that have ideal properties. The lower the total difference measure, the closer the match between the target molecule and the Sim-List molecule. The similarity searching process is arbitrarily cut off when the total difference measure

reaches 1000. This encompasses all the worthwhile column suggestions.

CHIRULE currently contains 96 molecules with 142 column references. These are spread over the 14 different columns contained in the Daicel Chiral Column Applications Guide[111]. The guide groups the columns according to the adsorbent used: cellulose ester (Wainer Type II), cellulose carbamates (Type II), ligand exchange (Type IV), crown ether (Type III), and polymethacrylate (Type III). Of these five, four had sufficient examples to allow them to be used for testing purposes. The column with the most examples for each adsorbent was chosen for use in testing. The four columns chosen to serve as test cases, shown in Table XIII, all have at least twice as many references as other columns found within the Chiral Column KB.

In testing a system such as this, the tests must be designed to use column types that are known to be successful for separating a significant fraction of the population of chiral molecules known to the system. This will insure that there are both 1) a reasonable number of columns and 2) a large number of molecules in the information space that the expert system is to probe. It would be unfruitful to test the system at either extreme: 1) where all the molecules were separated by one column, or 2) where each molecule was separated by only a single column. Summing the number of

molecules for each of the four column types shown in Table XIII reveals that one third of the chiral column references in CHIRULE's KBs are for other column types. This proportion should be sufficient to stress the methodology if the concepts upon which CHIRULE is based are not valid. The high success rates observed in Table XIII indicate that those concepts are, in fact, valid.

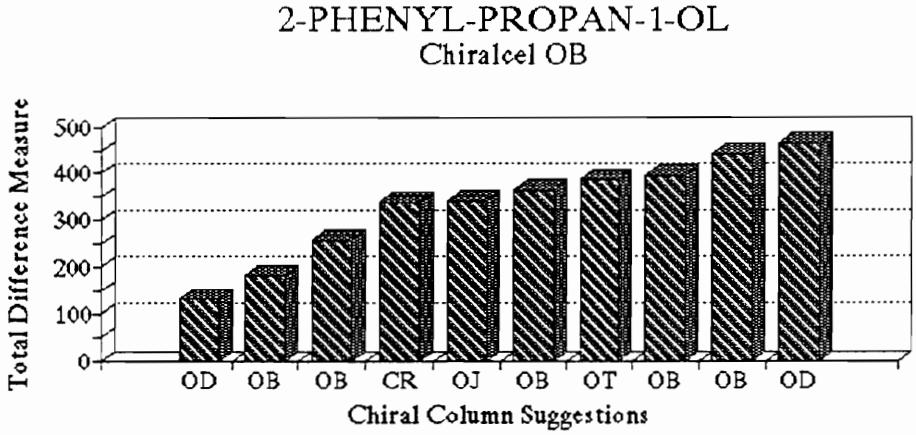
Looking more closely at Table XIII, the first column contains the name of the CSP used in the testing process. The second column shows the number of molecules having entries in the Chiral Column KB for that CSP. The third column depicts the success rate when only the very first CSP suggestion generated by CHIRULE is considered. The success rate is the percentage of CSP suggestions that are supported by information within the Chiral Column KB. These results proved encouraging. The less desirable response for the ChiralCel OB and OD columns (both cellulose columns) might suggest that other chemical properties than those employed in this work should be examined.

Expanding the domain of CHIRULE by incorporating the information of ChirBase is a logical extension of this work. The high success rates in Table XIII indicate that the four molecular properties chosen do in fact represent major aspects of the chemical environment experienced by the chiral carbon. As the size of CHIRULEs KBs grows, the

additional examples will further define the range of chemical environments unique to each CSP. This should result in a corresponding increase in the observed success rates. ChirBase's prohibitively high cost ($17,000) precluded its use in this project.

In a number of instances, the conditions depicted in Figure 31 were encountered. Note that lower total difference measure values are desirable since they represent a closer match between the target and Sim-List molecules. In these cases, the first CSP suggestion (Chiralcel OD) was unsupported by CHIRULE's KBs. However, the second CSP suggestion (Chiralcel OB) was supported and the total difference measures of the top two column suggestions differed by less than 100. In addition, the second column suggestion (Chiralcel OB) was repeatedly found in the lowest eight column suggestions. To represent these conditions, Table XIII also presents the success rate where the first or second column suggestion is supported. The high success rates prove the feasibility of (1) using various molecular properties to model the chemical environment of the molecule's chiral carbon, and (2) performing a similarity search to generate a list of chiral column suggestions.

Interestingly enough, utilizing each property individually did not yield universally acceptable results (see Table XIV). This is most likely a confirmation of the



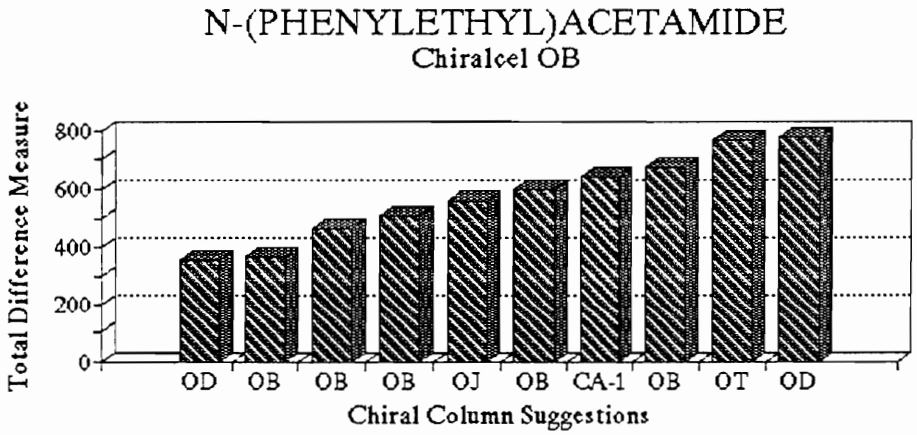


Figure 31: Investigating CHIRULE Test Data

Table XIV: CHIRULE Single Property Results

Property	CrownPak CR	ChiralCel OB	ChiralCel OD	ChiralPak WH
Polarizability	83%	24%	63%	67%
Alpha	*	*	*	*
Beta	96%	57%	42%	86%
Molecular Volume	77%	56%	63%	52%

*Alpha has been omitted because of the large number of similar values found within the Fragment Value KB.

initial speculation that no one property can accurately represent chirality. The most notable exception was that beta values (hydrogen bonding proton accepting component) seemed to be a fairly good predictor for Daicel CROWNPAK CR columns. The reporting of matches for alpha, the Hydrogen Bonding Proton Donating property, are omitted since for over a third of the molecular fragments the alpha values are zero.

In addition to testing CHIRULE against its own knowledge base, four "real life" molecules were used to investigate the chiral column predicting capabilities. A separations scientist[110] within the Chemistry Department at Virginia Tech provided a list of four compounds that he had been working with. Two of the compounds had been obtained from chemical suppliers (trifloro-anthral-ethanol and propranolol). The other two, phenyl-2-methyl-butanoate and (acetylamido) phenylalanine-methylester (AAPAME), had been synthesized within the department by Dr. Merola and Dr. Hanson respectively.

CHIRULE was asked to suggest a chiral column for the four molecules above. The chiral column suggestion with the lowest total difference measure agreed with the scientist's final selection in three of the four cases. The column that the scientist used for the fourth molecule, AAPAME, was also predicted, although with a higher total difference measure.

The scientist has indicated that the column he used did not yield baseline separation, indicating that there was room for improvement in the method. Unfortunately, the scientist left the University before the separation could be attempted with the column suggested by CHIRULE.

VIII.A.2. CHIRULE's Column Suggestion Methodology

CHIRULE's column recommendations are not limited to a specific CSP. The technique looks only at the chemical environment of the chiral carbon and available literature data in making its decisions. What limits the likelihood of suggesting any specific type of column is the number of reported separations that are contained within the knowledge base. If there are not as many examples of optically active molecules that can be separated by one particular column, than the likelihood of a new chiral molecule being chemically similar to a molecule separated by that column is reduced. Columns with more than 20 separation references were chosen for testing in CHIRULEs Chiral Column KB. It is felt that at least 20 references are necessary to adequately represent a particular column type.

From the discussion above, it can be inferred that this technique relies on a large number of known chiral separations being represented in the knowledge base. The more molecules that are available for comparison, the more

likely that a correct solution will be found. Such a large collection of information, for example, ChirBase[14] (see Chapter II.A.1) contains information on almost 17,000 chiral separations, and could easily be incorporated into the CHESS system. Note that both CHESS and ChirBase are founded on products from MDL[15] and use the same MOL file format, making their integration relatively easy.

CHIRULE is not bound by the molecular fragments that are currently contained in its KBs. As new fragments are encountered, SPARC is utilized to calculate the various property values required for the similarity search. This is in contrast to the approach taken by Berthod, Chang, and Armstrong[44]. Adding entries to their substituent energy lists involves performing tens of experiments to uncover the necessary regression coefficients.

The most beneficial use of CHIRULE could prove to be in the area of obtaining a better understanding of the separation mechanisms of chiral chromatography. In this limited study, the property adjustment values and normalization factors were implemented so that each molecular property contributed equally to the total difference measure. However, these factors could be varied and the resulting success rates utilized, in conjunction with an expanded KB of chiral separations, to investigate (1) what properties best describe the chemical environment

and (2) the optimum contribution of each of those properties towards each type of chiral column. Once researchers understand the contributions molecular properties make to the overall separation, this will undoubtedly lead to a better understanding of the separation mechanisms that underlie chiral separations.

VIII.A.3. Property Adjustment and Delta Values

Both the delta value and the property adjustment value affect the size and number of bins that the hashing function create. Before being hashed, each property value is adjusted so that its range of values is from 0 to 10. The delta value can be used to modify the number of bins that are created for all properties. The feasibility of the approach has been demonstrated using a delta value of 10. However, all of the predicates have been written in such a way as to allow experimenting with other delta and property adjustment values. It is believed the delta value will play a more important role in limiting the search domain as the size of the KB grows.

Another reasoning behind implementing variable delta values was to allow for the future investigation of having multiple delta values present in the Fragment Value Range KB. This would allow for the creation of multiple sets of bins for similarity searching, one set for each delta value.

These experiments would begin similarity searching with the smallest bins (largest delta). If an acceptable number of matches were obtained (i.e. 5-10), the process would stop there and print out the results. If not, the next smallest sets of bins (next largest delta) would be utilized and the experiment repeated.

VIII.A.4. CHIRULE's Total Difference Measure

There are several important items related to the calculated total difference measure. The normalizing factors described in Chapter V.C.4.b are employed so that each property has an equal contribution to the total difference measure. They are not related to either the property adjustment values which deal with generating the bins for the similarity search or the delta values described in the previous section.

In an effort to place further bounds on the similarity searching process, a restriction was placed on calculating the total difference measure between the target molecule and a Sim-List molecule. Each fragment pair must be found in the Fragment Match KB for at least one property. For example, if the target and Sim-List molecule had two fragments in common, the remaining two fragments would be subject to the similarity search process. In order to be found similar, each remaining fragment in the target

molecule must reside in the same hash bin as a fragment in the Sim-List molecule.

This restriction is accomplished by predicate "calc_diff" first determining the appropriate number of fragment pairs using the Fragment Match KB. Once the appropriate number of matching fragment pairs have been found, their total difference values are calculated, and summed into the total difference measure. This requirement helps to limit the breadth of the similarity search.

VIII.A.5. CHIRULE's Testing Procedures

There are a number of mechanisms for testing the validity of CHIRULE's suggestions. Text Frame 73 in Chapter VII.D.3 describes how the predicate "chirule" is used to query CHIRULE's KB for a column suggestion for the user's target molecule. The predicate "chirule" will prompt the user for the filename of the target molecule's MOL file, and requires that the molecule not exist in CHIRULE's KBs. If it is found to exist, the user will be informed and asked if they want to proceed.

Other predicates were developed to perform tests on a much larger scale. All of the testing predicates operate in a similar fashion to the predicate "chirule" except that they do not require the MOL file of the molecule be input. These predicates, shown in Table XV, can be divided into two

Table XV: CHIRULE Testing Predicates

Molecule Testing Predicates:

"tstcr ?name" - Test the CHIRULE KBs looking for molecules similar to the existing molecule ?name.

"testcr" - Test the CHIRULE KBs, prompting the user for the name of a molecule to use, and calling "tstcr" with that name to perform the test.

"test_all_sugg" - Explicitly call predicate "tstcr" with a number of different molecule names in order to test the CHIRULE KBs

Column Testing Predicates:

"test_col ?column" - Find all molecules that have a reported separation with column type '?column' and call "tstcr" to query CHIRULE for a suggestion.

"test_all_cols test_col" - Test each column type for all of the molecules using that column type. This is accomplished by repeatedly calling predicate "test_col" with each column type.

Other Predicates:

"list_mols ?column" - List out and count all of the molecules contained in CHIRULE's KBs that have references to chiral column '?column'.

"list_all_mols" - List out all of the molecules contained in CHIRULE's Molecule KB, and report the total number found.

"test_all_cols list_mols" - List out all of the molecules contained in CHIRULE's KBs grouping them by chiral column types.

"show_chiral ?name" - Display all of the information contained in CHIRULEs KBs for molecule ?name (molecule name, SMILES string, molecular fragments, fragment values for all properties, chiral column and reference)

"add2chiral" - Add a new molecule to CHIRULEs KBs. User will be prompted for the MOL file name of the molecule.

different categories: (1) those testing individual molecules and (2) those testing according to column types. Each of these categories rely on information that is already a part of CHIRULE's KBs. This is made possible by specifically removing the possibility of a direct match with itself when performing the similarity search. The molecule testing predicates list all of the column suggestions for a particular molecule. The column testing routines will test all molecules that can be separated by a particular column.

VIII.A.5.a. Testing by Molecule

The predicates described in this section focus on testing specific molecules. Predicate "tstcr" searches through CHIRULE's KBs to find similar molecules to the Target molecule name passed as a variable. The results consist of molecules that have been found to be similar to the target molecule. These results are displayed as a rank ordered list of column suggestions (see Text Frame 77) based on those similar molecules. The predicate "tstcr" is identical to the predicate "chirule" except that it does not have to read in the molecule's MOL file.

Predicate "testcr" prompts the user for the name of a single target molecule (rather than requiring a name as a variable). "testcr" then calls "tstcr" using the input target molecule name to query CHIRULE's knowledge base.

Another predicate, "test_all_sugg", consists of multiple calls to predicate "tstcr" each with a molecule name specified.

VIII.A.5.b. Testing by Column

In addition to testing specific molecules, it is advantageous to be able to test all molecules that are capable of being separated with a particular kind of column. Predicate "test_col" takes a particular type of column as a variable. It then chooses target molecules from the chiral Column KB that have a reported separation using that column. This is accomplished by calling predicate "testcr" with each target molecule name as they are found.

Invoking the predicate "test_all_cols test_col" (i.e. using the variable "test_col") will repeatedly call the predicate "test_col" with a number of different column types. This predicate allows for testing CHIRULE's predicting capabilities for all column types. Predicate "test_cols", rather than testing all column types, tests only the four columns that were used to judge the success of CHIRULE (Daicel CrownPak CR, ChiralCel OB, ChiralCel OD, and ChiralPak WH). Predicates "test_cols" and "test_all_cols" were used to generate the results discussed below.

VIII.A.5.c. Other Testing Procedures

There are several other pieces of information that are useful in evaluating the CHIRULE system. These issues concern determining the molecules and columns that are currently contained in CHIRULE's KBs, as well as ways to expand that information.

Predicate "list_mols" will list out and count all of the molecules in CHIRULE's KBs that can be separated by the column supplied as a variable. Calling the predicate "test_all_cols list_mols" will take a list of columns and display and count all of the molecules that can be separated by those columns. Another predicate, "list_all_mols" will list out and count all of the molecules in CHIRULE's Molecule KB. These predicates were used to tabulate the range of knowledge contained within CHIRULE's KBs.

There are two additional predicates that interact with all of CHIRULE's KBs. The first is predicate "show_chirule", which will display all of the information for a molecule name given as a variable contained within CHIRULE's various KBs (name, SMILES string, molecular fragments, fragment values for all properties, chiral column, and reference). The second is predicate "add2chiral" which is used to add additional molecules to CHIRULEs KBs. This predicate prompts the user for the MOL file name for the molecule to input into CHIRULE, and then

adds all of the appropriate information to the various KBs. It differs from predicate "chirule" in that it does not perform a similarity search to determine a column suggestion.

VIII.A.6. Future Directions

Similarity searching was the approach taken in proving the feasibility of the chiral column suggestion system CHIRULE. Any one Sim-List molecule can be successfully separated using a number of different column types. Hence when a Sim-List molecule is found to be "similar" to a target molecule, all of the Sim-List molecule's columns are equally suggested as starting points in the development of a new method. A method for preferentially selecting one of the available column types would provide additional assistance to the user.

Another logical extension of this work would be to incorporate the separation mechanism of the individual columns (as described in Chapter II.A.3) into the total difference measure calculation. The exact mechanism for implementing this will involve evaluating, correlating, and understanding statistical analyses of each properties contribution to chirality. A large knowledge base of known separations would be essential to the success of such a statistical analysis. Discovering the influence a

particular property has for a type of column would be a valuable tool as researchers continue to investigate the interactions that comprise a chiral separation.

Buda and Mislow's[50] work based on Hausdorff's chirality measure, as described in Chapter II.A.5, is based on the belief that "chirality is independent of a molecules physical and chemical manifestations, being tied rather to the geometry of a molecule." This implies that some molecules are more chiral than others. It is this authors belief that the chemistry of a molecule is also important, and that shape is just another parameter that can be used to describe chirality. Experimenting with this and other properties will further improve the success rates experienced by CHIRULE as the properties used to describe the chiral carbon's chemical environment are enhanced.

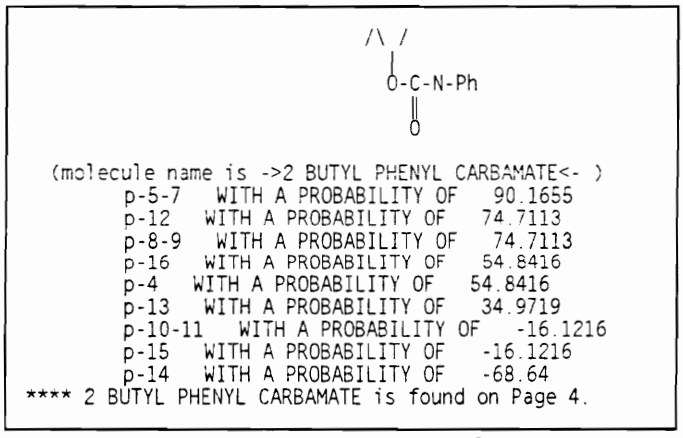
VIII.B. OPTIMER Results

OPTIMER development was undertaken to demonstrate that, given the correct set of knowledge acquisition tools, it is possible for a scientist[110] to develop an expert system. OPTIMER models the decision tree knowledge found in most chiral column manufacturers selection guides. A number of issues arose during the development and testing of OPTIMER that warrant further discussion.

VIII.B.1. Discussion of Results

Figure 32 through Figure 39 show the results of the OPTIMER tests. For OPTIMER, larger numbers are better since the value represents the probability that the molecule is found on that page number. In every test case (except for 2-butylphenylcarbamate which is discussed below), the page number that was reported with the highest probability was correct. These tests indicate that OPTIMER successfully represents the decision tree knowledge found in the applications guide.

There was an unexpected outcome of the OPTIMER system. The molecule 2-butylphenylcarbamate was posed as a query. OPTIMER responded that the molecule should be found on pages 5-7 of the guide, although the molecule was actually taken from page 4 (see Text Frame 79). Further inspection uncovered that the molecule was incorrectly placed on page 4 within the application guide. According to the decision tree (see Figure 7), the third question to be asked regards the presence of a hydroxy group. In order to be placed on page 4, there must be a



Text Frame 79: Unexpected OPTIMER Result

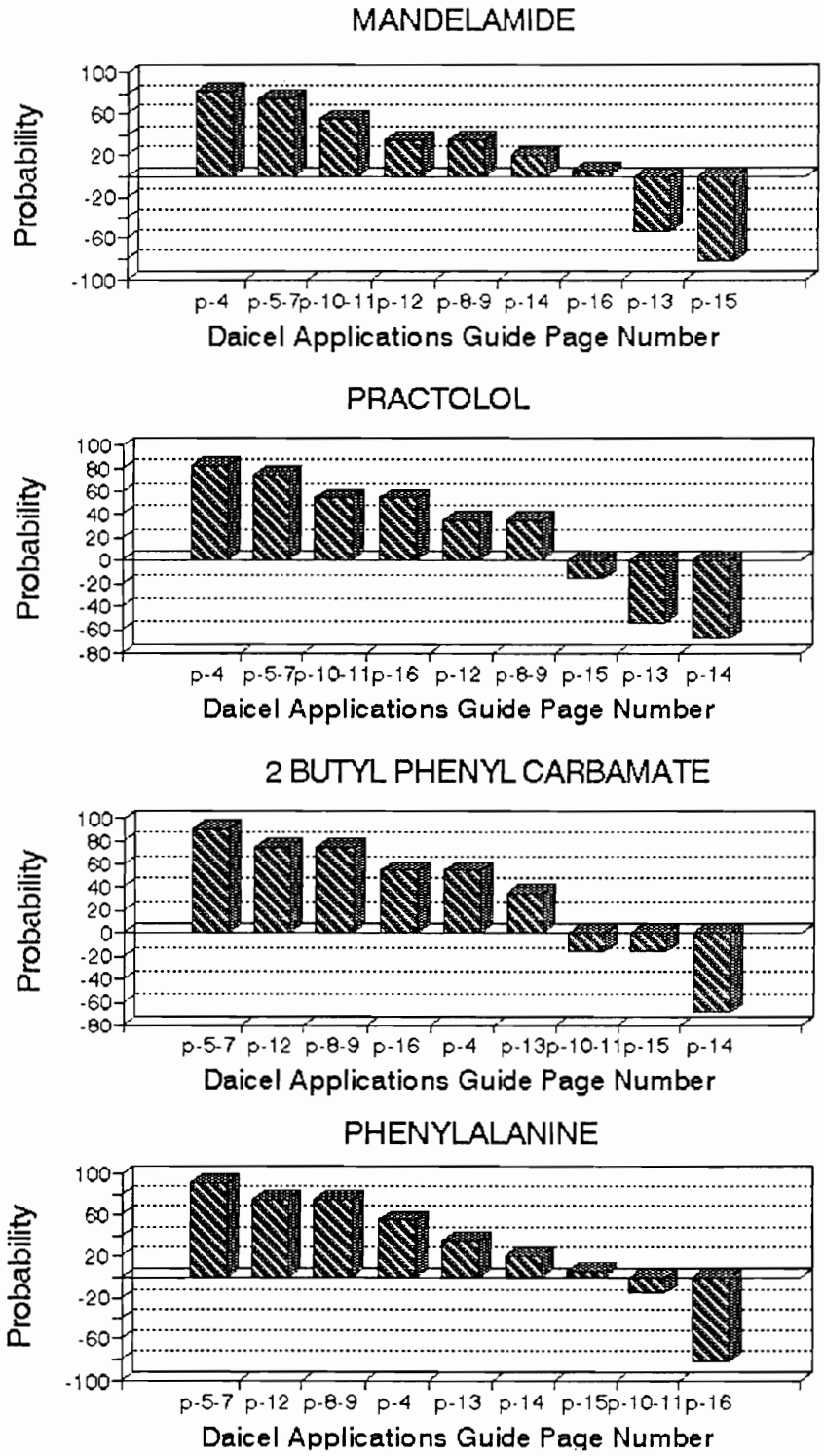


Figure 32: OPTIMER Results, Group 1

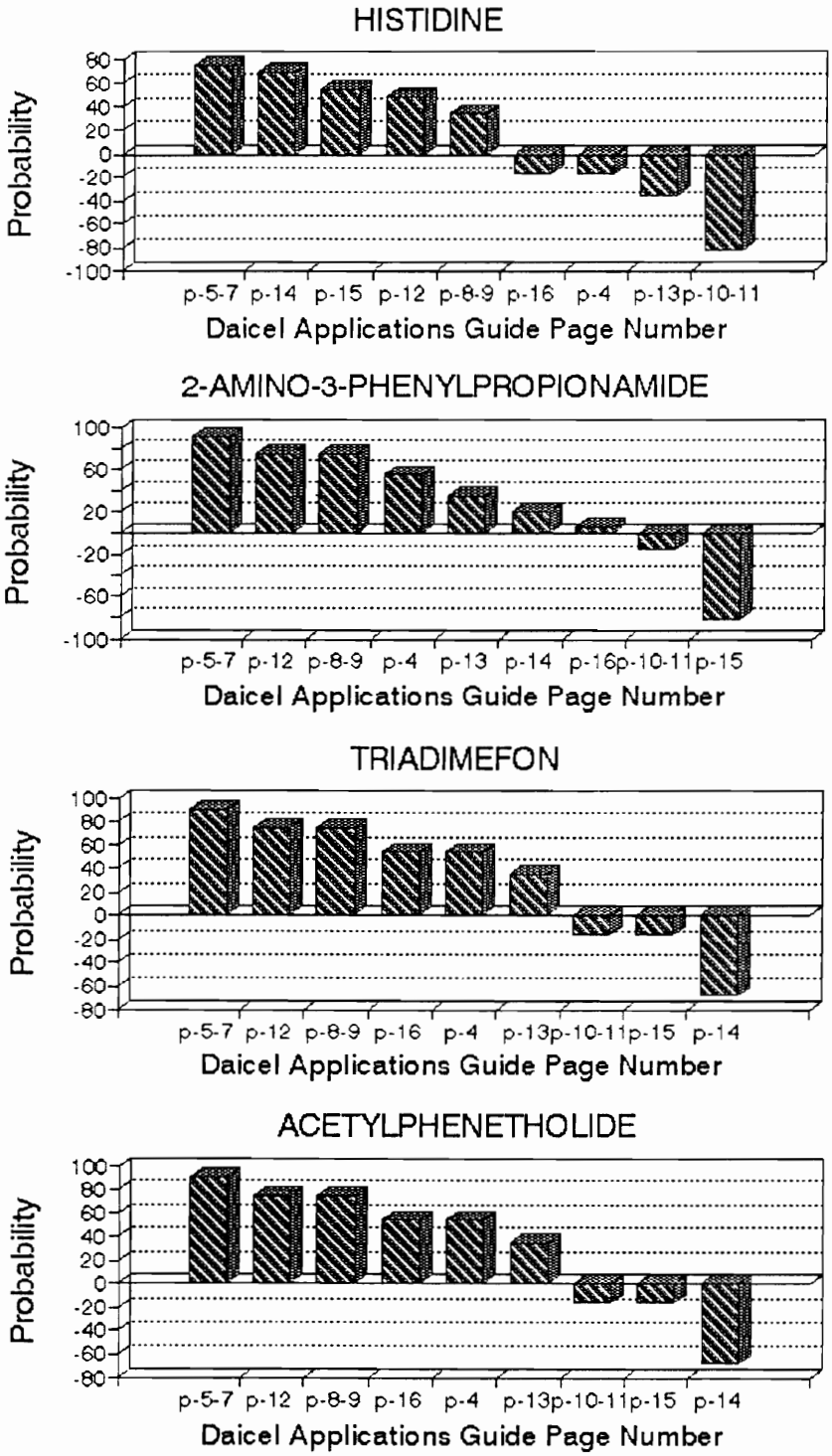


Figure 33: OPTIMER Results, Group 2

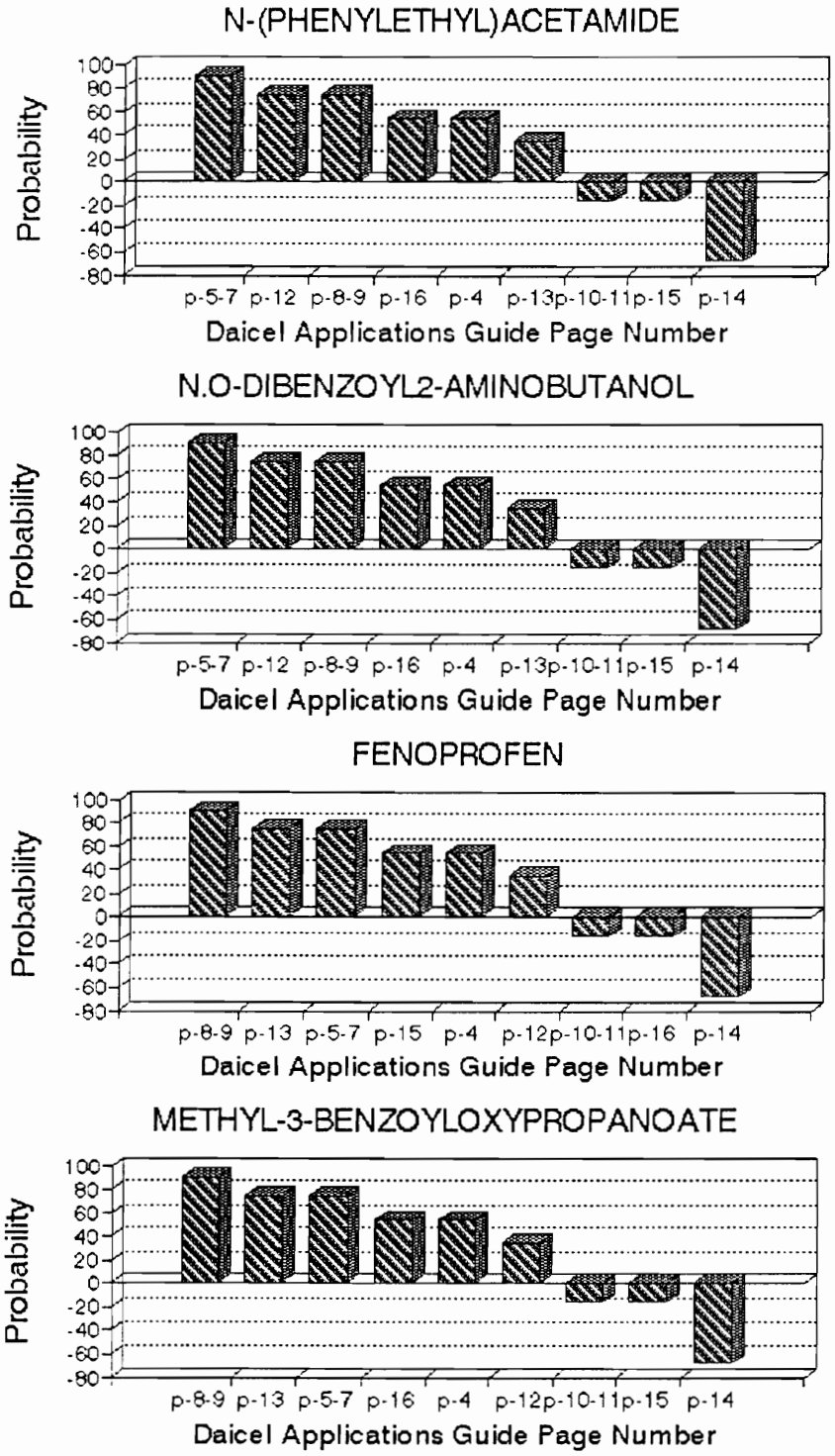


Figure 34: OPTIMER Results, Group 3

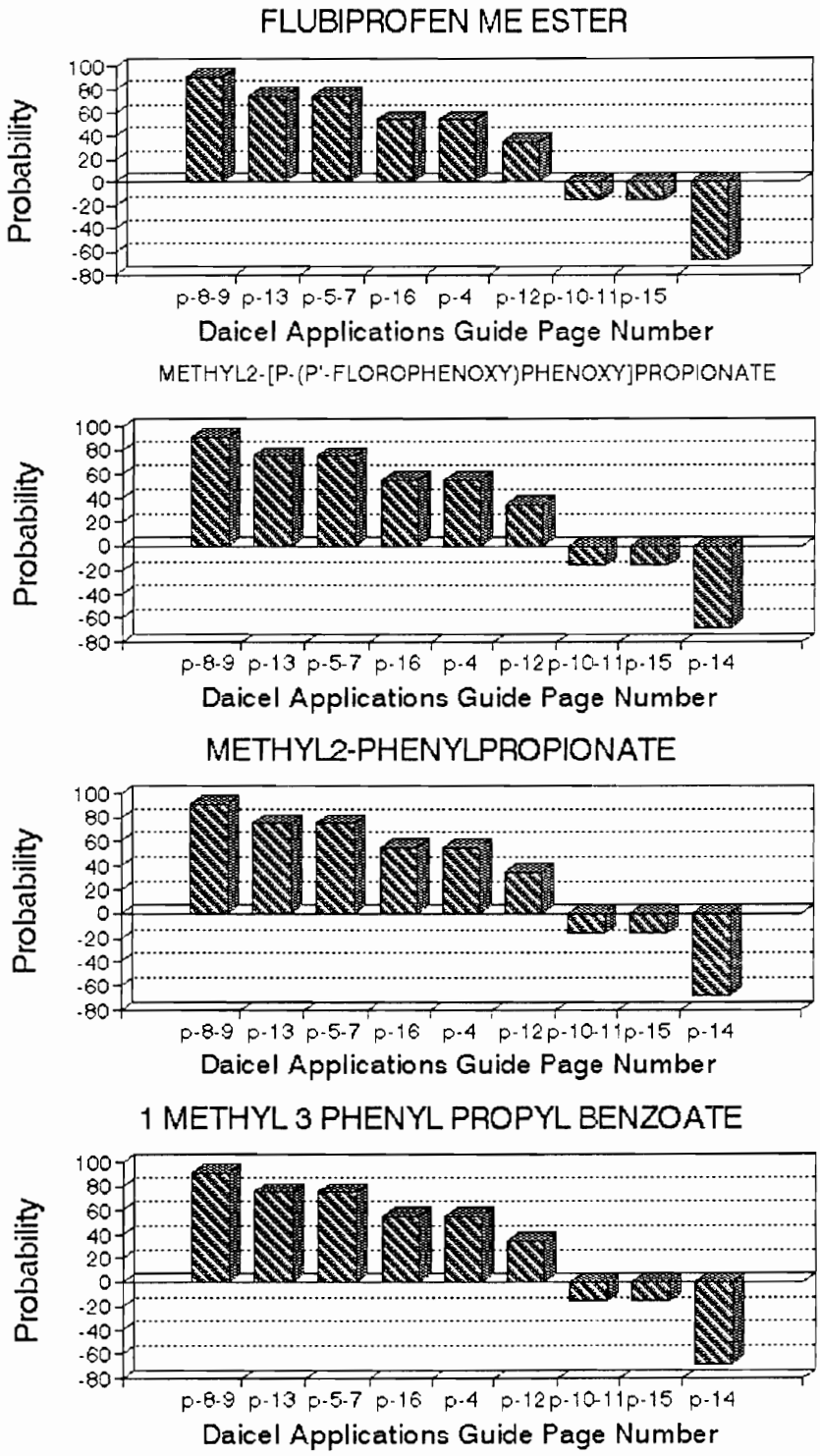


Figure 35: OPTIMER Results, Group 4

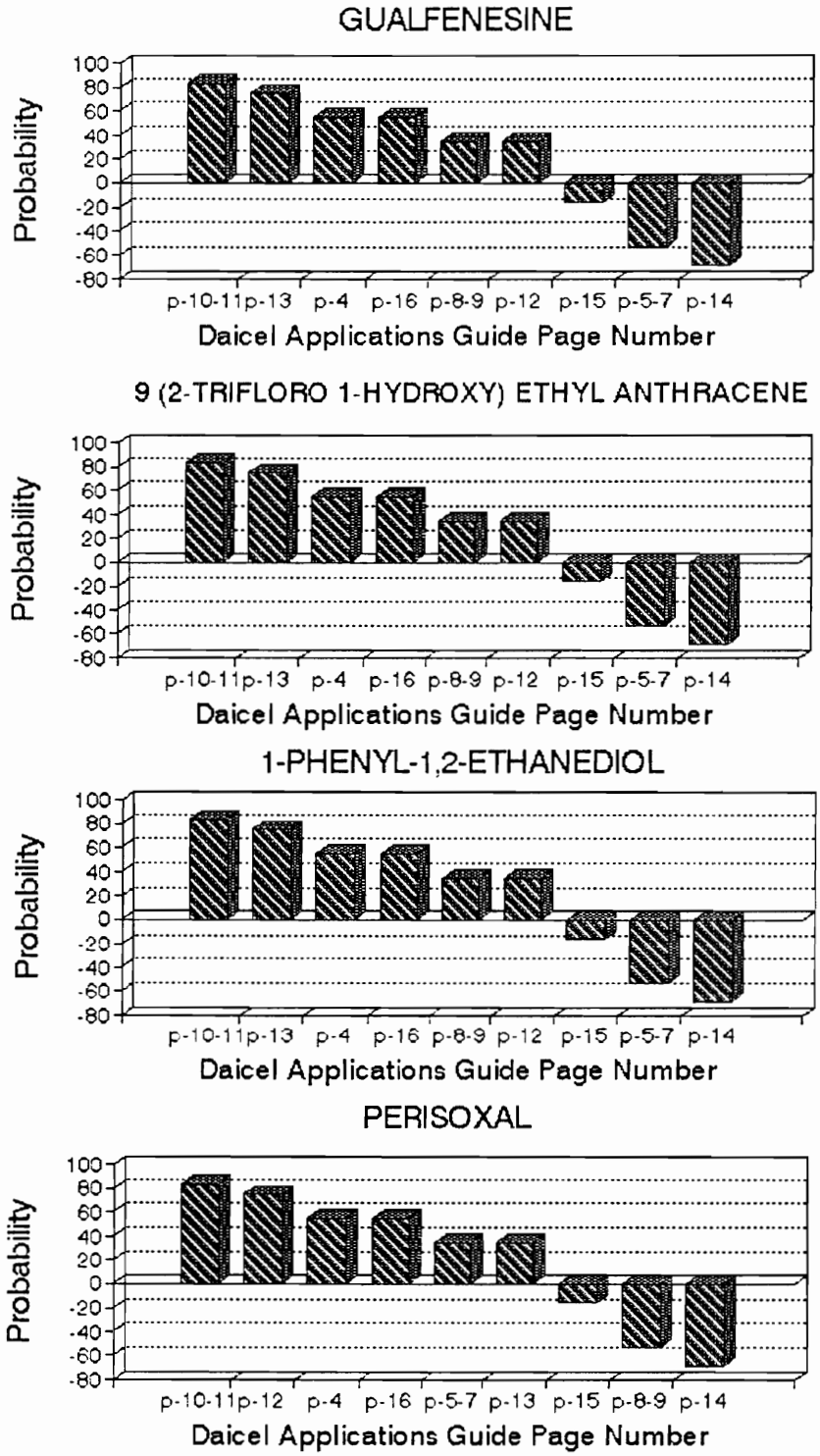


Figure 36: OPTIMER Results, Group 5

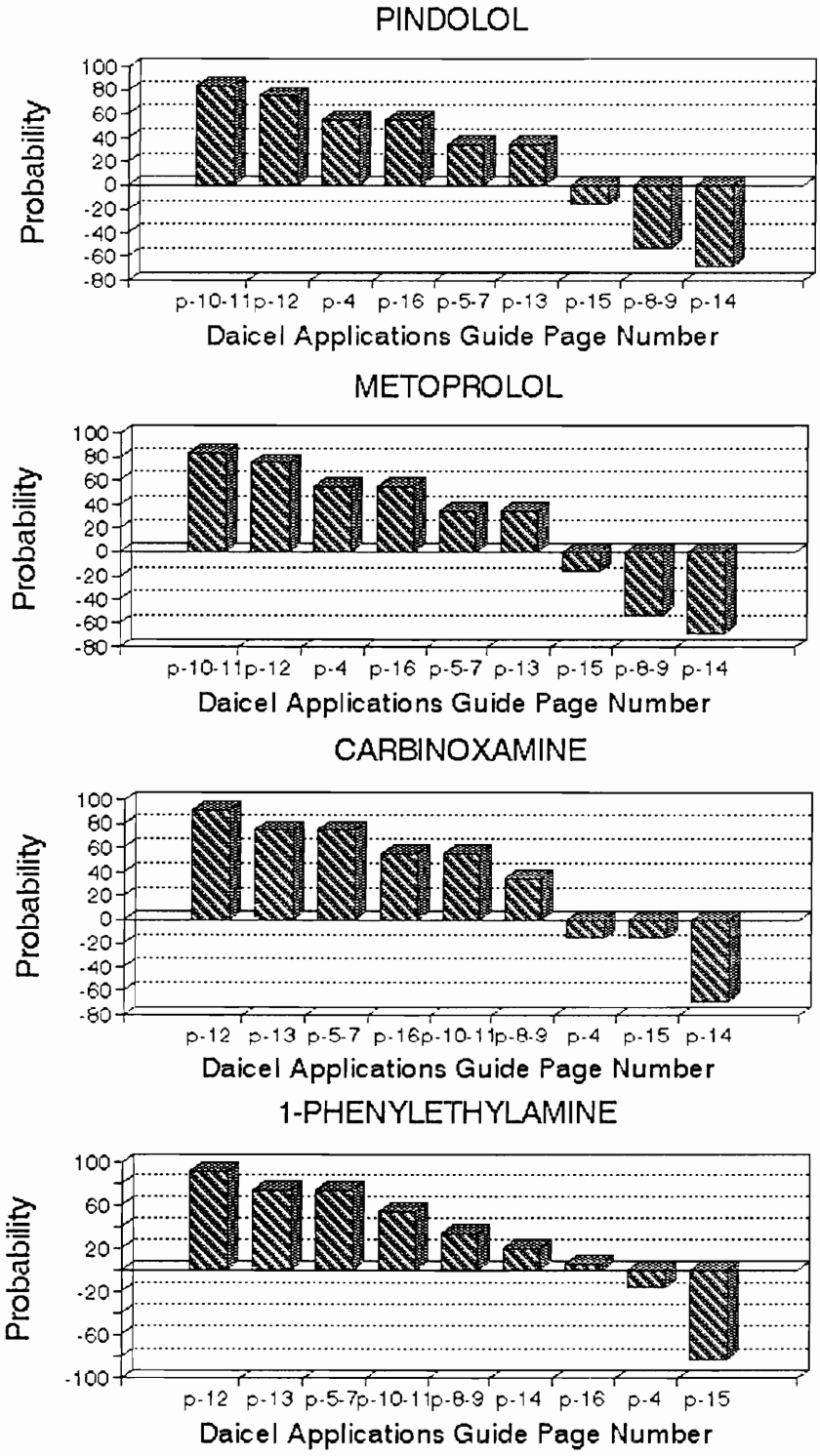


Figure 37: OPTIMER Results, Group 6

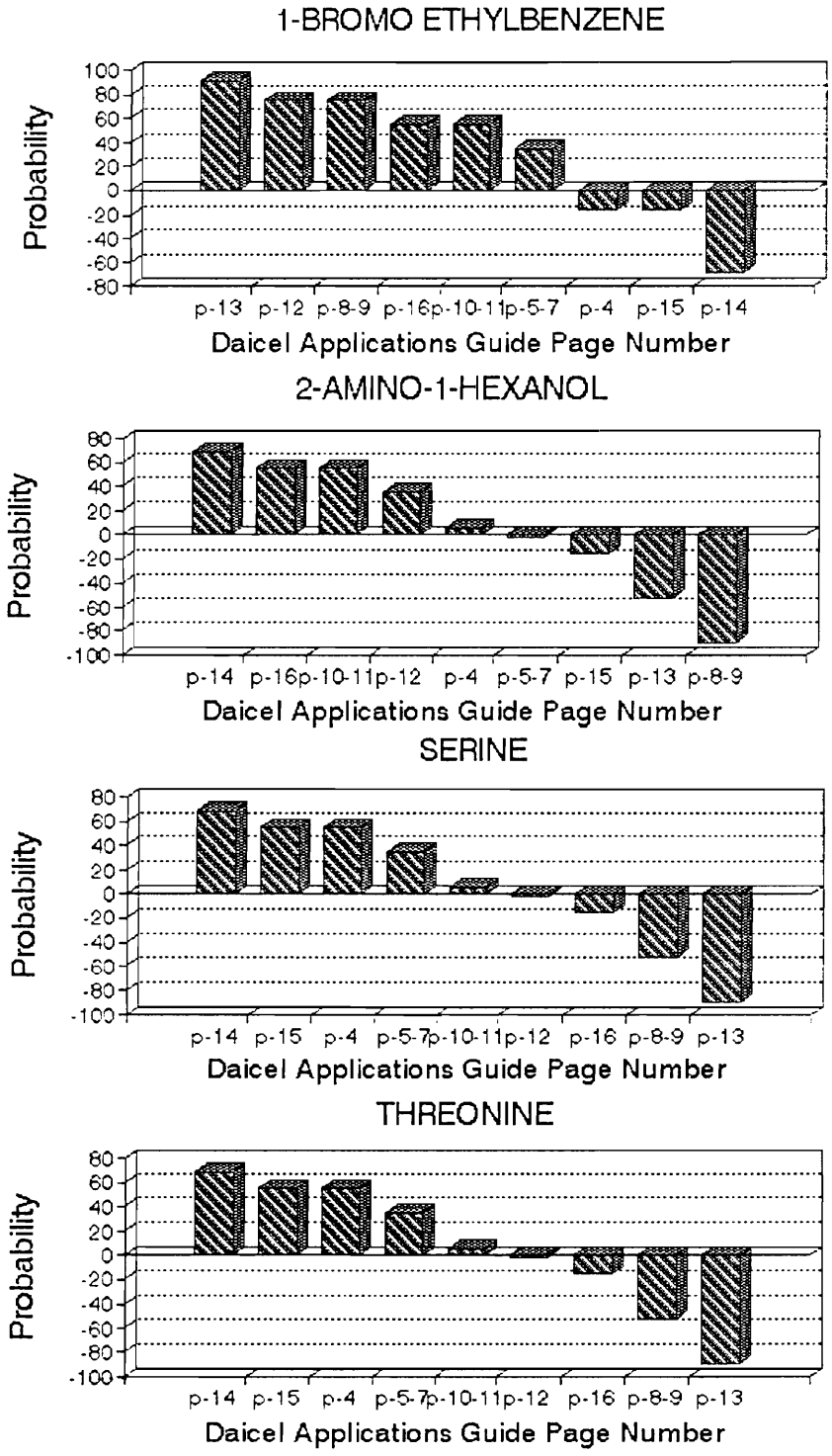


Figure 38: OPTIMER Results, Group 7

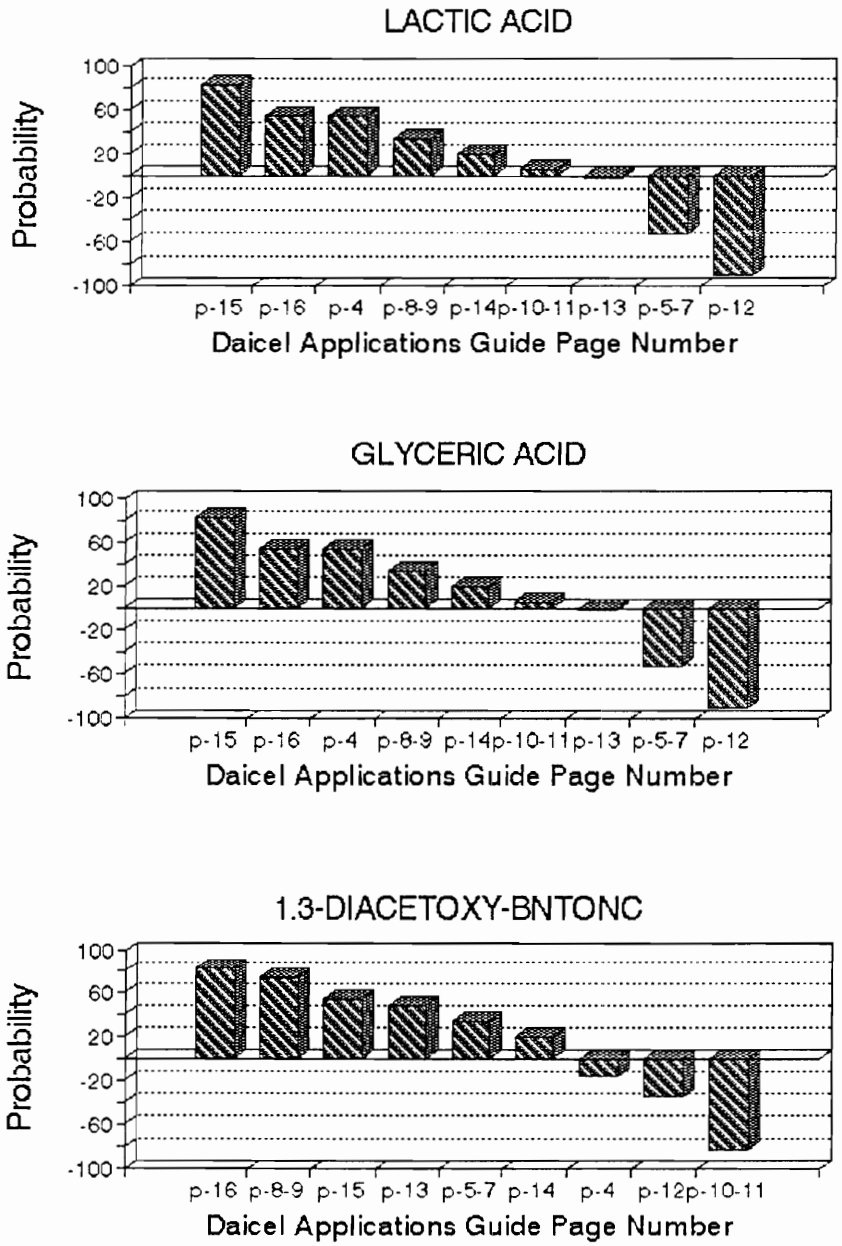


Figure 39: OPTIMER Results, Group 8

hydroxy group present. However, 2-butylphenylcarbamate does not contain a hydroxy group. According to the decision tree, 2-butylphenylcarbamate should reside on page 5-7.

VIII.B.2. Testing Procedures

The testing of the OPTIMER system consisted of consulting the Daicel Chiral Column Application Guide[111], and randomly picking several molecules from each of the pages. These molecules were input as queries to determine the page number that OPTIMER would give as the highest probability of where the molecule was located. The results of such a query are shown in Text Frame 28. Figure 32 through Figure 39 show the results of the show graphical representations of the results from these data runs. Predicates "test1_rb" through "test4_rb" were developed to query the OPTIMER system for a series of specific molecules.

VIII.B.3. Future Directions

Chapter IV.B describes how choosing the specific trait names shown in Table III allowed KEY (and hence OPTIMER) to take advantage of the hooks built into the CHESS shell. These predicates determine the presence (or absence) of functional groups within the user's target molecule. In demonstrating KEY's feasibility, it was not necessary to provide a mechanism for the development of new queries

through the linking of existing predicates and rulebases. Expanding KEY to include these capabilities would simply be a trivial exercise in software engineering. This section outlines details on how to implement this linking of predicates and rulebases.

For example, a developer is working on an application and wishes to base a decision on the presence of an amide. One way to accomplish this would be to write a Prolog predicate similar to the one shown in Text Frame 80. However, if the developer did not have the necessary tools available to pursue this avenue, the same result could be attained through having KEY link the results of the two functional group queries "has_CdO" and "has_N3". Each of these queries would be used as trait names, however the developer would append "&1" to the front of each indicating that KEY was to link the two queries using a variable (variable "1"). An additional instance of predicate "exrule" (see module KEY.HC in Appendix E) would be developed to handle the linking of traits.

```
((has_amide ?name ?A1 ?A2) if
        (molecule ?name ? ?atlist ?bdlist)
        (has_CdO ?name ?A1 ?A3)
        (has_N3 ?name ?A2 ?)
        (findsb ?bdlist ?A1 ?A2)
  )
```

Text Frame 80: Prolog Predicate for Determining Amides

The new "exrule" predicate would be executed if an ampersand "&" were used as the first character of any trait. After encountering the ampersand, if this were the first

linking trait encountered that used variable "1", the central atom number returned from the execution of the trait would be stored internally in the variable "1". If it was the second linking trait encountered using the variable, it would execute the second trait, and then determine if the two central atoms were in fact connected. Although a more rigorous definition would be required to exclude less commonly encountered functional groups, this example does demonstrate the avenues available for expanding the utility of CHESS and KEY.

In addition to allowing the linking of existing predicates to develop additional predicates from within KEY, plans have been developed that would allow the linking of rulebases to allow researchers to build on the strengths of previously developed systems. For example, researcher "A" develops the rulebase depicted in Text Frame 81 for choosing the color best suits their disposition. The answers provided by this system (the elements) would simply be colors of the rainbow. The traits used to distinguish between them would be warm/cold, dark/bright, and other emotional and psychological issues. Running this system would involve the user commenting on their personality (warm or cold, dark or

Rulebase 1:
Choosing A Favorite Color

Elements: green, red, white, blue,...

Traits: dark/bright, warm/cold,...

Text Frame 81: Color Rulebase

bright outlook, etc). The system would suggest what color best "fits" them.

Another researcher, "B", wants to develop a rulebase to predict the best automobile for an individual (see Text Frame 82). BMW, Porsche, Miata, Rolls Royce, Ferrari would be examples of this systems elements. Appropriate traits would include the engine size, passenger seating, color, and craftsmanship. However, researcher "B" learns of the color rulebase developed by "A", and wants to make use of its expertise to avoid selecting a color that seems inappropriate (a red Rolls Royce, for example). Researcher "B" would take advantage of the color rulebase by linking it into the car selection rulebase.

Rulebase 2:
Choosing An Automobile

Elements: Porsche, Miata, BMW, Rolls Royce....

Traits: engine size, passenger seating, color*, craftsmanship....

*link to existing color rulebase.

Text Frame 82: Automobile Rulebase

The linking of predicates would be accomplished by allowing another rulebase name to be given as a trait name by placing the "at" symbol "@" in front of the rulebase name. An additional instance of the predicate "exrule" would detect any trait with an "@" symbol as the first character, then call the predicate "run_rb" using the rulebase specified. The predicate "run_rb" would have to be

modified so that it asserted the top one (or two) results generated from executing the rulebase.

The functional group hooks built into the CHESS shell could be linked to other rulebases. The hooks operate by having the query succeed if the functional group is present, and fail if absent. Since any KEY rulebase will always execute, a different mechanism for judging the success of a rulebase query must be implemented. The result of running a KEY rulebase is the system will return a list of responses, called elements, rank ordered according to a probability value. Remember that PCT utilizes a spreadsheet like structure to regularize an experts knowledge: the columns are the elements, and represent the answers that are being sought while the rows are the traits which are used to distinguish between the elements. The response with the highest probability is considered to be the "correct" answer.

In order to determine if the rulebase has been successfully run, the answer (or element) that is expected must be provided. This way, if the rulebase is run and the response matches the expected answer, the rulebase is considered to succeed. This requires that the trait name include the expected result of executing the rulebase as part of the trait name. Using OPTIMER as an example, the trait name "@chiral-b=p4" would only succeed if the rulebase

"chiral-b" were found to return the element "p4" as the response with the highest probability.

VIII.C. System Integration Issues

The contributions towards system integration contained within this work should not be overlooked. Several researchers are beginning to look at the integration of various programs to solve problems[33]. The researchers noted that the choice of exclusively using the PC platform resulted in a number of interesting difficulties. Memory constraints, difficulty writing embedding programs, and the need to modularize to speed up execution times were some of the obstacles encountered. However, through judicious use of hardware and software platforms, these obstacles were overcome. The power of the resulting integrated system in both maintainability and clarity is self evident. The development of complex expert systems through the integration of a number of smaller, simpler systems holds much promise.

This work proposed that a distributed computing approach to expert systems programming will resolve a number of the issues currently encountered by researchers. CHIRULE relies on the computer network to draw together the results from various computer platforms into one working system. This strategy allows the application to take advantage of

the strengths of those platforms without being restricted by their limitations. The success of CHIRULE and OPTIMER demonstrates the validity of using a multi-windowing environment on a personal work station that serves as the central node of such a distributed system.

VIII.D. A Concluding Analysis

Being able to build on the work of other researchers in an area is imperative in preventing what is commonly referred to as "re-inventing the wheel". The author realizes that Prolog is not as commonly used in programming as BASIC or Pascal. However, it is felt that the judicious use of a distinctly "English-like" syntax allows other researchers to be able to build on what has already been accomplished. Using the pre-defined structure queries in Table VI along with the examples shown in Table III and Table V in Chapter III, researchers should be able to construct new functional group searching procedures. This will allow the further development of many of the issues raised by this work.

Other researchers are proposing the introduction of other types of knowledge, such as chemometrics, into chemical expert systems[33]. It is their hope that its inclusion will help analytical chemists to develop a better understanding of chemometrics and enable them to utilize the

technology. Future expert system shells will undoubtedly incorporate these capabilities as well.

IX. Conclusion

The development of a general purpose expert system shell capable of reasoning in the chemical arena is an ambitious goal. CHESS is a general, versatile, configurable tool developed by the author. The ability to symbolically manipulate functional groups, and the reasoning structure that enables the integration of the various knowledge bases found within the shell are among the unique features of this Chemistry-Based Expert System Shell. CHESS's potential has been demonstrated by the success of the two specific systems the author constructed using its tools: CHIRULE and OPTIMER.

CHIRULE is an expert system that will suggest a chiral stationary phase to serve as the starting point in the development of a method for separating enantiomeric species. CHIRULE generates an N-dimensional information space, where each dimension represents a different molecular property, and each point in the space represents a successful separation of enantiomers by a particular CSP. Known enantiomeric separations are entered into CHIRULE's knowledge base by calculating a value for each of the molecular properties. The values are then used to plot a point for that separation within the N-dimensional space. This information space can be utilized in two different ways: 1) in suggesting a column to use in developing a

method for separating an "unknown" pair of enantiomers, and 2) in studying the separation methods of different chiral stationary phases.

Rather than using properties of the entire molecule to describe the chemical environment at the chiral center, the molecule is fragmented around the chiral carbon. Each of the four fragments are then treated separately. Calculating the molecular properties on each fragment provided a way to model the chemical environment experienced by the chiral carbon. This model can then be used to implement a mechanism for targeting a chiral column for use with an "unknown" enantiomeric pair.

To suggest a column for a target "unknown" molecule the molecular properties used to develop the information space are calculated for each of the molecular fragments around the target's chiral carbon atom. These values are then used to place the target molecule inside the information space. Known separations that are found to be close to the target molecule are deemed to have chiral carbons with chemically similar environments, and hence should both be successfully separated with the same CSP.

There are several situations where this model of a chiral carbon's chemical environment begins to degrade. When there is a low population of any particular type of chiral column the information space does not contain enough

points to provide an accurate mapping for an unknown. For example, when a particular type of chiral column has only one reported separation in the information space, the likelihood of an unknown being "similar" to it is very small.

Conversely, if the information space does not contain a variety of chiral column types, the model could degrade into a statistical fit on one column that could not be used to describe a majority of chiral separations. This could occur if examples of only one or two chiral column types were included in the information space. Both of these situations were avoided by choosing a few examples from a large number of column types (14 in all), and then including a number of examples for a few specific types (20 some examples each for 4 of the columns). This strategy has proven to be effective.

The success of CHIRULE indicates that molecular properties such as molecular polarizability, hydrogen bonding, and molecular volume can be used as vehicles for exploring the chemical environment of the molecule's chiral carbon. Now that the proposed model has proven to describe some aspects of the chemical environment of a molecule's chiral carbon, the information space itself can be used for studying the different chiral stationary phases. Future research incorporating other properties in constructing the

information spaces should assist in the understanding of the mechanisms behind the chiral separation process itself. The resulting data clusters in the information spaces will show the molecular properties that are important to the chiral column's separation mechanism. This would lead to a better understanding of the underlying mechanisms of the various types of chiral columns.

Other aspects of CHESS have also proven to be a success. OPTIMER demonstrates that a scientist, given the proper tools and a well bounded problem, can effectively develop an expert system. KEY, the automated knowledge extraction utility, plays a vital role in providing a mechanism for transferring and regularizing the scientists knowledge into an expert system knowledge base.

Although product integration was only a small part of this thesis, its application here demonstrates what can be accomplished by combining currently available packages with newly developed programs. Such integration will become much easier if the industry standardizes on one particular operating environment (such as Microsoft Windows). With all the pitfalls of such environments, the advantages of inter-process communications (DDL, dynamic data links) do lend credibility to arguments in their favor.

X. Addendum

Few dissertations or the papers derived from them are usually tested or extended. Yet, that is the true test of their worth. This is usually done by others following up on leads pointed out in the original work. In this instance, the author was fortunate to experience this process himself. Between completing the dissertation and defending it, the author was given the opportunity to test the concept against the French chiral data base, Chirbase, in the possession of The United States Pharmacopeial Convention, Inc. (USP).

Drs. R. Lindauer and R. Frankewich at USP are investigating the use of chiral separations as a way to update a number of their monographs concerning optically active pharmaceutical drugs. They are interested in any technology that would help to reduce the development time required to formulate the necessary methods. Thus they are interested in both Chirbase and CHIRULE.

Chirbase is a conventional database that now contains approximately 21,000 examples of chiral separations. It runs under MDL's Chembase on a 486/66MHz PC. This gave the opportunity of testing two concepts simultaneously: (1) the possibility of extending the expert system CHIRULE to include other knowledge bases, and (2) examining whether the

96 compounds described in the dissertation were indeed a statistically satisfactory test bed.

CHIRULE makes CSP suggestions not through traditional database queries, but through a similarity search based on molecular properties. A clear understanding of the data presented for CHIRULE in the dissertation is imperative to discerning the significance of the experiment performed at USP. The reader is reminded that Table I in the introduction to the dissertation lists the terminology used in describing the results. Table XVIa reproduces the information found in Chapter 8, Table XIII of the dissertation. The tests consist of submitting molecules (target molecules) to CHIRULE that have reported methods employing chiral stationary phases (CSPs). CHIRULE responds with a list of CSPs it feels are appropriate for use in developing a separation method for the target molecules. Recall that for testing purposes, the similarity search specifically rules out reporting results associated with the target molecule itself within CHIRULEs knowledge bases (KBs). This would be nothing more than a simple database lookup and would not exercise the chemical reasoning capabilities of CHIRULE.

CHIRULE rank orders the list of CSP suggestions according to the CSP's total difference measures. Each CSP suggestion is arrived at by relating the target molecule to

Table XVI: CHIRULE/Chirbase Results

Table XVI.a - ORIGINAL CHIRULE TESTING RESULTS:

Chiral Column (CSP)	Number of Molecules using CSP	First CSP Suggestion Supported	First or Second CSP Supported
CrownPak CR	29	90%	90%
ChiralCel OB	22	77%	95%
ChiralCel OD	22	64%	73%
ChiralPak WH	23	83%	87%
TOTALS:	96	79%	88%

Table XVI.b - UPDATED RESULTS AFTER CHIRBASE EXPERIMENT AT USP:

Chiral Column (CSP)	Number of Molecules using CSP	First CSP Suggestion Supported	First or Second CSP Supported
CrownPak CR	29	93%	93%
ChiralCel OB	22	82%	95%
ChiralCel OD	22	77%	77%
ChiralPak WH	23	96%	96%
TOTALS:	96	88%	91%

a molecule within CHIRULEs KB (SimList molecules). CHIRULE analyzes the chiral carbon's chemical environment in both target and SimList molecules and calculates the total difference measure for corresponding fragments. If the environments are determined to be similar, then both molecules should interact similarly with the same. The CSP reported for the SimList molecule should also separate the target molecule. **Lower total difference measures are better** as they represent a closer match between the Target molecule and the SimList molecule (actually a closer match between the two chiral carbons' chemical environments).

The first CSP suggestion made by CHIRULE was compared to the target molecule's known CSP, resulting in the first column of results in Table XVIa. However, in a number of cases, the resulting rank ordered list of CSPs indicated that the second CSP should also be considered (see Figure 31). Consideration of the first two CSPs yielded the second column of results shown in Table XVIa.

The USP experiment involved those molecules for which CHIRULE's first CSP suggestion did not correspond to the known separation strategy (see Table XVII and Figure 40). The dissertation was careful to point out that these cases should not be viewed as either "correct" or "incorrect", but rather that the first CSP suggestions were "unsupported by CHIRULE's knowledge bases". The MOL files of these

Table XVII: Molecules used in Chirbase Experiment

	CHIRULE/DAICEL TEST		CHIRBASE
Known CSP Separation and Molecule Name	First CSP Suggestion	Second CSP Suggestion	Query Result
CROWNPAK CR (7%⁻)			
1) 2-amino-1-phenyl ethanol	ChiralCel OK	ChiralCel OB	Neither
2) 1-phenylethyl amine	ChiralCel OB	ChiralCel OB	Neither
3) L-isoleucine	ChiralPak WE	ChiralPak WM	ChiralPak WE
CHIRALCEL OB (4%⁻)			
4) 2-acetoxy-3-phenyl propane	ChiralCel OD	ChiralCel OB	*
5) 2-phenyl-propan-1-ol	ChiralCel OD	ChiralCel OB	ChiralCel OD
6) methyl-3-benzoyloxy propanoate	ChiralPak OT	ChiralCel OB	*
7) n-(phenylethyl) acetamide	ChiralCel OD	ChiralCel OB	*
8) 3-phenylbutyl benzoate	ChiralCel OJ	ChiralPak OT	Neither
CHIRALCEL OD (23%⁻)			
9) mandelic acid	ChiralCel OB	ChiralCel OB	Neither
10) 2-phenyl propionic acid	ChiralCel OB	ChiralCel OB	Neither
11) 2-butylphenyl carbamate	ChiralCel OB	ChiralCel OB	Neither
12) gualfenesine	ChiralCel OB	CrownPak CR	Neither
13) 1(1-hydroxyethyl) naphthalene	ChiralCel OB	ChiralCel OD	ChiralCel OB
14) 9(2 trifloro-1-hydroxy) ethyl anthracene	ChiralCel OJ	ChiralCel OJ	ChiralCel OJ
15) carbinoxamine	ChiralPak OT	CrownPak CR	Neither
16) N-CBZ-asparagine	ChiralPak WH	ChiralPak WH	ChiralPak WH
CHIRALPAK WH (4%⁻)			
17) mandelic acid	ChiralCel OB	ChiralCel OB	Neither
18) histidine	CrownPak CR	CrownPak CR	CrownPak CR
19) L-methionine	CrownPak CR	ChiralPak WH	CrownPak CR
20) L-isoleucine	ChiralPak WE	ChiralPak WM	ChiralPak WE
"REAL WORLD"			
21) (acetylamido) phenylalanine methylester	ChiralCel OB	ChiralPak WH	ChiralCel OB

⁺ Percentage of unresolved CSP suggestions to the total population of each CSP: CrownPak CR - 2/29, ChiralCel OB - 1/22, ChiralCel OD - 5/22, ChiralPak WH - 1/23.

* CHIRBASE contains Daicel data, and therefore must support ChiralCel OB.

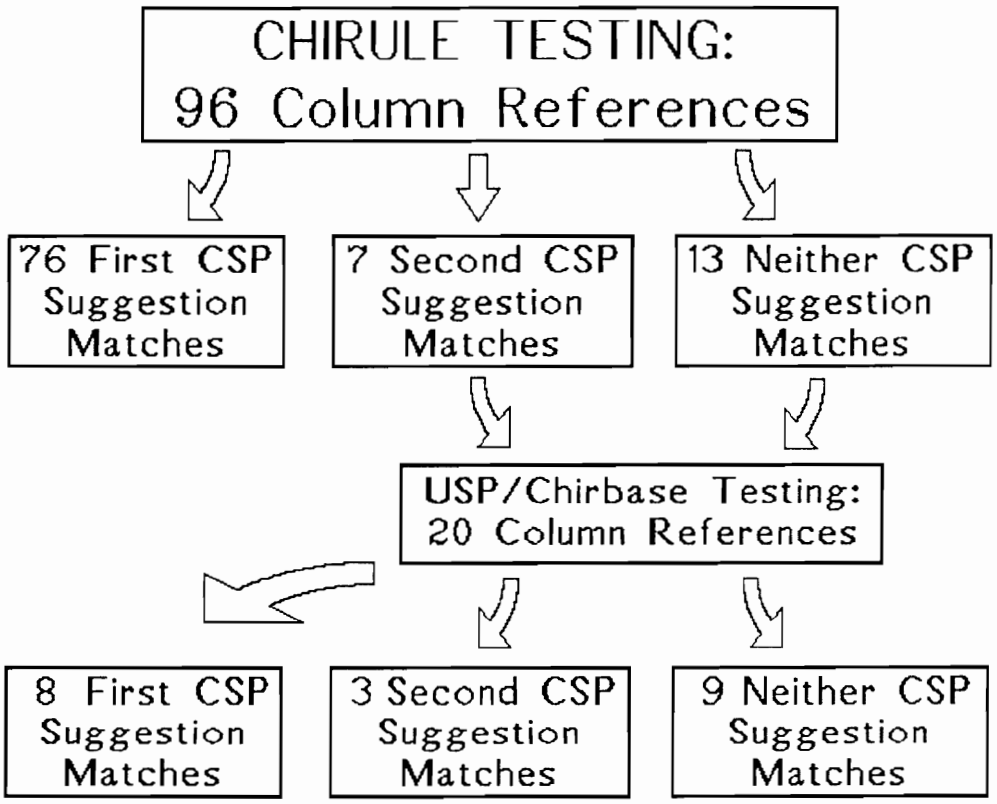


Figure 40: Testing Scheme for CHIRULE

molecules, twenty (20) in all, were extracted from CHIRULE and directly posed as queries to Chirbase. Although this was done manually because of the hardware platform available, the "hooks" are available to make the process one seamless automatic operation. The chiral column suggestions returned by Chirbase were then compared with those of CHIRULE.

In almost 50% of the cases submitted, Chirbase CONFIRMED THE FIRST CSP CHOICE made by CHIRULE (as shown in Table XVII). CHIRULE's first CSP choice was indeed a column which is reported in the literature to be a satisfactory separation system, despite the fact that such information was NOT explicitly included in the knowledge base. Indeed the suggested chiral column was not the CSP type undergoing testing. The results of this new experiment on the search success rates are shown in Table XVIb. It is easy to see that using a larger, literature supported data base increased the success rate based on comparing CHIRULE's first CSP, or first+second CSP suggestions with Chirbase's results. Both types of success rates increased, and actually began to converge. The best convergence is shown for three column types.

Several factors are obviously necessary for a convergence that could begin to reach 100% success rates. First, it is not unusual for misleading data to be contained

in limited compilations of data such as the Daicel Applications Guide. In many instances researchers do not rigorously determine what the BEST column is for a particular molecule. Work often ceases when a somewhat satisfactory solution is achieved. Chirbase, with its larger size, and more catholic input, removes some of this bias. However, it is also probably incomplete.

Second, our understanding of chiral separations is currently incomplete, and many believe that different column types involve different separation mechanisms. It is suggested that the exact convergence shown in Table XVIb for CrownPak CR, Chiralcel OD, and ChiralPak WH substantiates the conclusion that the CHIRULE similarity search approach and parametric weightings are valid for these systems. The experiment with Chirbase confirmed that CHIRULE's first CSP suggestions are indeed valid. Simple extrapolations explain the two apparent dichotomies involving ChiralCel OB and ChiralCel OD. In the former, the second rank hits are often the correct solution. For the molecules in question (1) the total difference measure for the second hit is below 400, (2) the total difference measures of the first and second rank hits differ by less than 20, and (3) the second CSP suggested also appeared in five or more of the top ten (10) CSP suggestions. This indicates the parametric choices and weightings for ChiralCel OB are "almost" correct. The

interested reader is referred to cases 5, 13, 16, and 20 in Table XVII where these assumptions are clearly supported.

The new experiment still leaves 9 molecules for which proper column references are not obtained (see Figure 40). Over half of these are associated with a particular column type, ChiralCel OD. This chiral column also has the lowest success rate (77%). This suggests that a different mechanism of separation may be involved, and that further statistical studies, augmented by a more extensive empirical data base, could result in not only a different choice of parameters and their weightings, but a clearer picture of chiral separation theory.

In addition to these tests, CHIRULE was also queried with four "real world" molecules that a separations scientist here at Virginia Tech had experience with. The dissertation reported that CHIRULE was able to confirm three of the four CSPs used by the scientist. While the remaining molecule, (acetylamido) phenylalanine methylester, was separated on a ChiralCel OD column, baseline separation was not achieved. Chirbase was queried for this molecule, and reported the ChiralCel OB column also suggested by CHIRULE. Here, baseline separation could have been achieved had CHIRULE been available to the scientist at the time of the exeperiment.

In conclusion, these new results demonstrate several things. First, the use of a larger data base does increase the overall hit rate from 88% to 91%. This supports the conclusion that a more extensive support base would improve an already acceptable search strategy. On the other hand, the small improvement supports the conclusion that the 96 compound library used originally was a statistically significant test bed.

Improvement beyond the 90th percentile will require a better understanding of chiral separations. Empirical knowledge coupled with correlations derived from statistical evaluation of parametric variations readily available from CHIRULE should assist in reaching these desirable and necessary goals.

REFERENCES

1. "FDA issues flexible policy on chiral drugs" *Chemical and Engineering News*, 15 June 1992, 70, 24, 5.

2. Taylor, D.R.; Maher, K. *J Chromatogr. Sci.* March **1992**, 30, p67-85.

3. Ahuja, Satinder, ed. "Chiral Separations by Liquid Chromatography" ACS Symposium Series 471, **1991**, p1-26.

4. Blaschke, G.; Kraft, H.P.; Fickentscher, K.; Kohler, F. "Chromatographic racemic separation of thalidomide and teratogenic activity of its enantiomers", *Arzneim.-Forsch.*, **1979**, 29, 1640-42.

5. "Top 200 Drugs in 1982", *Pharmacy Times*, April **1982**, p25.

6. Okamoto, Y. *CHEMTECH* March **1987**, p176.

7. Pasteur, L. *Ann. Chim. Phys.* **1848** 24, 442-59.

8. Armstrong, D.W. *Anal. Chem.*, **1987**, 59, 2, 84A.

9. Buda, A.B.; Auf der Heyde, T.; Mislow, K. *Angew. Chem., Int. Ed. Engl.*, in press. The author wishes to thank Dr. Mislow for providing a pre-print to review.

10. Van't Hoff, J.H. *Arch. Meerl., Sci Exactes Natur.* **1874**, 9, 445-54.

11. Le Bel, J.A. *Bull. Soc. Chim. Paris* **1874**, 22, 337-47.

12. Dalgliesh, C.E. "Optical resolution of aromatic amino acids on paper chromatograms" *J. Chem. Soc* **1952**, 47, 3940-42.

13. Pirkle, W.H.; Finn, J.M.; Schriner, J.L.; Hamper, B.C. *J. Am. Chem. Soc.* **1981**, 103, 3964-66.

14. Roussel, C.; Piras, P. "CHIRBASE Project", ENSSPICAM, Faculté des Sciences Saint-Jérôme, F 13397 Marseille Cedex 13, (FAX (33) 91 02 77 76).

15. ChemBase, ChemTalk, ChemDraw, and MACCS are products of Molecular Design Limited (MDL), 2132 Farallon Drive, San Leandro, California 94577, (800) 635-0064.

16. Pirkle, W.H.; Pochapsky, T.C. "Considerations of chiral recognition relevant to the liquid chromatographic separation of enantiomers" *Chem. Rev.* **1989** 89, 347-62.

17. Wainer, I.W. *Tr. in Anal. Chem* **1987** 6, 5, 125-34.

18. Wainer, I.W.; "A Practical Guide to the Selection and Use of HPLC Chiral Stationary Phases", J.T. Baker, Phillipsburg, New Jersey, **1988**.

19. Armstrong, D.W. "Optical Isomer Separation by Liquid Chromatography" *Anal. Chem.* **1987** 59, 2, A84-91.

20. Armstrong, D.W.; Hilton, M.; Coffin, L.; "Multimodal Chiral Stationary Phases for Liquid Chromatography" *LC-GC*, **1992**, 9, 9(September), 646-652.

21. Däppen, R.; Arm, H.; Meyer, V.R. "Applications and Limitations of Commercially available Chiral Stationary Phases for HPLC" *J. Chromatography*, **1986**, 373, 1-20.

22. Snyder, L.R.; Glajch, J.L.; Kirckland, J.J.; *Practical HPLC Method Development*, John Wiley & Sons, Inc.; New York, 1988.

23. Chapter 8, equation 8.1 in Reference 22.

24. Peeters, A.; Buydens, L.; Massart, D.L.; Schoenmakers, P.J.; "An Expert System for the Selection of Criteria for Selectivity Optimization in High-Pressure Liquid Chromatography" *Chromatographia* **1988**, 26, 101-109.

25. Bach, R.; Karnicky, J.; Abbott, S. "An Expert System for High Performance Liquid Chromatography Methods Development" in reference 98, 278-296.

26. Hindriks, R.; Maris, F.; Vink, J.; Peeters, A.; De Smet, M.; Massart, D.L.; Buydens, L.; "Expert System for the Selection of Initial High-Performance Liquid Chromatographic Conditions for the Analysis of Pharmaceuticals" *J. Chromatography* **1989**, 485, 255-265.

27. Karnicky, J., Personal Communication.

28. Williams, S.S.; "Application of expert systems programming to high-performance liquid chromatography method development" *TRAC* **1990**, 9(2), 63-65.

29. Williams, S.S.; Excoffier, J.-L.; Abbot, S.; presented at *Pittsburgh Conference, Atlanta, Ga, 1989,* abstract No. 1162.

30. Goulder, D.; Blaffert, T.; Blokland, A.; Buydens, L.; Chhabra, A.; Cleland, A.; Dunand, N.; Hindriks, H,; Kateman, G.; van Leeuwen, H.; Massart, D.; Mulholland, M.; Musch, G.; Naish, P.; Peeters, A.; Postma, G.; Schoenmakers, P.; de Smet, M.; Vandeginste, B.; Vink, J.; "Expert Systems for Chemical Analysis (ESPRIT Project 1570)" *Chromatographia* **1988**, 26, 237-243. Adapted from Figure 1 with permission.

31. Musch, G.; De Smet, M.; Massart, D.L. *J. Chrom.* **1985**, 348, 97-110.

32. Musch, G.; Massart, D.L. *J. Chrom* **1986** 370, 1-19.

33. Conti, P.; Hamoir, T.; De Smet, M.; Piryns, H.; Vanden Driessche, N.; Maris, F.; Hindriks, H.; Schoenmakers, P.J.; Massart, D.L. "Integrating expert systems for high-performance liquid chromatographic method development" *Chemometrics and Intelligent Laboratory Systems* **1991** 11, 27-35.

34. De Smet, M.; Peeters, A.; Buydens, L.; Massart, D.L.; "Expert system for the selection of high-performance liquid chromatographic methods in pharmaceutical analysis" *Journal of Chromatography* **1988**, 457, 25-42.

35. De Smet, M.; Musch, G.; Peeters, A.; Buydens, L.; Massart, D.L.; "Expert system for the selection of high-performance liquid chromatographic methods for the analysis of drugs: validation of the rules for the selection of the mobile phase" *Journal of Chromatography* **1998**, 485, 237-253.

36. Hindriks, R.; Maris, F.; Vink, J.; Peeters, A.; De Smet, M.; Massart, D.L.; Buydens, L.; "Expert system for the selection of initial high-performance liquid chromatographic conditions for the analysis of pharmaceuticals" *Journal of Chromatography* **1989**, 485, 255-265.

37. Peeters, A.; Buydens, L.; Massart, D.L.; Schoenmakers, P.; "An expert system for the selection of criteria for the selectivity optimization in high pressure liquid chromatography" *Chromatographia* **1988**, 26, 190-195.

38. KES, Knowledge Engineering System, Software Architecture and Engineering, Arlington, VA, USA. KES is available for both PCs and Apollo work stations.

39. Buydens, L.M.C.; van Leeuwen, J.A.; Mulholland, M.; Vandeginste, B.G.M.; Kateman, G.; "An expert system for the validation of high-performance liquid chromatographic methods" *TRAC* **1990**, 9, 2, 58-62.

40. Gant, J.R.; Vandemark, F.L.; Poile, A.F.; "A Method Development System for LC" *American Laboratory* **1990**, May, 15-23.

41. DRYLAB I/plus, DRYLAB G/plus, and DRYLAB GC are products of LC Resources Inc., 2930 Camino Diable, Suite 110, Walnut Creek, CA 94596, (510) 930-9043.

42. Snyder, L.R.; Dolan, J.W.; Lommen, D.C.; "Drylab Computer Simulation for High-Performance Liquid Chromatographic Method Development: I. Isocratic Elution" *Journal of Chromatography*, **1989**, 485, 65-89.

43. Dolan, J.W.; Lommen, D.C.; Snyder, L.R.; "Drylab Computer Simulation for High-Performance Liquid Chromatographic Method Development: II. Gradient Elution" *Journal of Chromatography*, **1989**, 485, 91-112.

44. Berthod, A.; Chang, S.C.; Armstrong, D.W. *Anal. Chem.* **1992**, 64, 395-404.

45. Lipkowitz, K.B.; Baker, B. "Computational Analysis of Chiral Recognition in Pirkle Phases" *Anal. Chem.* **1990**, 62, 770-774.

46. Lipkowitz, K.B.; Demeter, D.A.; Zegarra, R.; Larter, R.; Darden, T. "A Protocol for Determining Enantioselective Binding of Chiral Analytes on Chiral Chromatographic Surfaces" *J. Am. Chem. Soc.* **1988**, 110, 3446-52.

47. Lipkowitz, K.B.; Baker, B.; Zegarra, R. *J. Computational Chemistry* **1989** 10(5) 718-732.

48. Mislow, K.; Siegel, J. *JACS* **1984** 106, 3319.

49. Topiol, S.; Sabio, M.; Moroz, J.; and Caldwell, W.B.; "Computational Studies of the Interactions of Chiral Molecules" *J. Am. Chem. Soc.* **1988**, 110, 8367-76.

50. Buda, A.B.; Mislow, K *JACS*, **1992**, 114, 6006-6012.

51. Kelvin, W.T. in Clay, C.J. "Baltimore Lectures on Molecular Dynamics and the Wave Theory of Light", London, **1904**, 619.

52. Stauffer, S.T.; Dessy, R.E.; Roach, J.W.; "CHESS, a Chemistry-Based Expert System Shell" in Reference 99, 34-48.

53. Wiswesser, W.J. "A Line-Formula Chemical Notation" Crowell, New York, **1954**.

54. Weininger, David; "Smiles, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules" *JCICS* **1988**, 28, 31-36.

55. Wipke, W.T.; Dyott, T.M.; "Simulation and Evaluation of Chemical Synthesis. Computer Representation and Manipulation of Stereochemistry" *JACS* **1974**, 4825.

56. Barnard, J.M.; *JCICS* **1990**, 30, 81-96.

57. Bebak, H.; et. al. "The Standard Molecular Data Format (SMD Format) as an Integration Tool in Computer Chemistry" *JCICS* **1989**, 29, 1-5.

58. Gasteiger, J.; Hendriks, B.M.P.; Hoever, P.; Jochum, C.; Somberg, H. "JCAMP-CS. A Standard Exchange Format for Chemical Structure Information in Computer Readable Format" (unpublished material).

59. MACCS v5.1x, running on a VAX 8800 with the Aldrich database of over 32,000 compounds.

60. Esack, A.; Bersohn, M. "A Program for Rapid and Automatic Functional Group Recognition" *J.C.S. Perkin I* **1974** 2463-70.

61. Esack, A.; Bersohn, M. "A Program for Rapid and Automatic Functional Group Recognition" *J.C.S. Perkin I* **1974** 2463-70.

62. Armstrong, J.L.; Hibbert, D.B. *JCICS* **1989**, 29, 2, 51-60.

63. Borman, Stu "New QSAR Techniques Eyed For Environmental Assessments" *Chemical and Engineering News* 19 February 1990, 68, 8, 20-23.

64. Hunter, R.S.; Culver, F.D.; "QSAR SYSTEM USER MANUAL, A Structure-activity Based Chemical Modeling and Information System", A Joint Project of: The U.S. Environmental Protection Agency, Environmental Research Laboratory, Duluth, Minnesota and Montana State University, Institute for Biological and Chemical Process analysis (IPA), Bozeman, Montana; August 1988.

65. Russom, C.L.; Anderson, E.B.; Greenwood, B.E.; Pilli, A.; "ASTER: an integration of the ACQUIRE data base and the QSAR system for use in ecological risk assessments" *The Science of the Total Environment* **1991** 109, 110, 667-670.

66. CAChe Scientific, P.O. Box 500 M/S 13-400, Beaverton, Oregon 97077, (503) 627-3737.

67. Richards, N. "Using Computer-Aided Chemistry to Predict Chemical Structures and Properties" *Scientific Computing & Automation* July 1992, 8, 8, 33-37.

68. McDaniel, V.K.; "SPARC USERS MANUAL, Prototype Version 1.0" Environmental Research Laboratory, U.S. Environmental Protection Agency, Athens, GA.

69. Maggiora, G.M.; Johnson, M.A. "Introduction to similarity in Chemistry" in reference 70, 1-13.

70. *Concepts and Applications of Molecular Similarity*, Johnson, M.A., Maggiora, G.B., Eds.; John Wiley & Sons, Inc.: New York, 1990.

71. Bawden, David "Application of Two-Dimensional Chemical Similarity Measures to Database Analysis and Querying" in reference 70, 65-76.

72. Workshop on Similarity in Organic Chemistry, sponsored by the Beilstein Institute, held in Bozen (Bolzano), Italy in May of 1992, as reported in *Journal of Chemical Information and Computer Sciences*, **1992**, vol. 32, no. 6.

73. Hart, A.; *Knowledge Acquisition for Expert Systems*, McGraw-Hill Book Company, New York, 1986.

74. Michaelsen, R. H.; Michie, D.; Boulanger, A. *Byte*, **1985**, 10(4), 303-313.

75. Schlieper, W.A.; Isenhour, T.L.; Marshall, J.C.; "Using Analytical Data to Build Expert Systems" *JCICS* 1988, 28, 159-163.

76. Kelly, G.A. "The psychology of Personal Constructs", Norton, New York, **1955**.

77. Shaw, M.L.G., Ed., "Recent Advances in Personal Construct Technology" Academic Press, New York, **1981**, pp147-182.

78. Intelligence Ware, Inc., 9800 S. Sepulveda Boulevard, Suite 730, Los Angeles, California 90045, (213) 417-8896.

79. Boeing Computer Services, Advanced Technology Center for Computer Sciences, P.O. Box 24346, MS 7L-64, Seattle, Washington 98124-0346, (206) 865-3241, contact: Arthur T. Nagai.

80. Boose, J.H. "Expertise Transfer for Expert System Design" Elsevier, New York, **1986**.

81. Boose, J.H.; "Use of Repertory Grid-Centered Knowledge Acquisition Tools for Knowledge-Based Systems" In *2nd. AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop* Banff, October, **1987**.

82. Davis, R.; Lenat, D.B. "Knowledge-Based Systems in Artificial Intelligence" McGraw-Hill, New York, **1982**.

83. Roach, J.W. "Robot Problem Solving Using an Incomplete Model", *IEEE Computer Graphics and Applications* **1983**, p154.

84. Wade, J.W. "Practical Charting Techniques" McGraw-Hill, New York, **1969**.

85. Cathey, D.L. "Constructing an Automated Visual Design Consultant Defining the Problem Domain", M.S. Thesis, Department of Computer Science, Virginia Polytechnic Institute and State University, **1986**.

86. Reddy, Jayachandra Lakshmana; "SITE: System for Interactive Transfer of Expertise", Thesis, Department of Computer Science, Virginia Polytechnic Institute and State University, **1989**.

87. Report entitled "Expert Systems" submitted in Chemistry 5130 by Scott T. Stauffer on December 16, 1985.

88. Duda, R.O.; Shortliffe, E.H.; "Expert Systems Research", *Science* 15 April 1983, 220, 4594, p.261.

89. Michaelsen, R.H.; Michie, D.; Boulanger,A.; "The Technology of Expert Systems", *Byte* April 1985, p303.

90. Dessy, R.E. ,ed.; "Expert Systems Part II", *Anal. Chem.* **1984**, 56, p1312A.

91. Nau,D.S.; "Expert Computer Systems", *Computer* **1983**, 16, 2, p.63.

92. Dessy, R.E., ed.; "Expert Systems. Part I" *Anal. Chem.* **1984**, 56, 1200A-1212A.

93. Van Leeuwen, J.A.; VanDeginste, B.G.M.; Postma, G.J.; Kateman, G.; "An Evaluation of Expert System Tools and their Use in High-performance Liquid Chromatography" *Chemometrics and Intelligent Laboratory Ssytems* **1989**, 6, 239-252.

94. Settle, F.A., Jr.; Pleva, M.A.; "Expert system development tools for chemists" *Chemometrics and Intelligent Laboratory Systems* **1991**, 11, 13-26.

95. Hushon, J.M. Ed.; "Expert Systems for Environmental Applications, American Chemical Society Symposium Series Vol. 431" American Chemical Society, Washington, DC, **1990**.

96. Hushon, J.M. "Overview of Environmental Expert Systems" in Reference 95, 1-24.

97. Yurman, D. "Success Factors for Expert Systems" in Reference 95, 25-38.

98. *Artificial Intelligence Applications in Chemistry, American Chemical Society Symposium Series Vol. 306*, Pierce, T.H., Hohne, B.A. Eds.; American Chemical Society, Washington, DC, **1986**.

99. *Expert System Applications in Chemistry, American Chemical Society Symposium Series Vol. 408*, Hohne, B.A., Pierce, T.H., Eds.; American Chemical Society, Washington, DC, **1989**.

100. Massart, D.L.; Hopke, P.K. *JCICS* **1985**, 25, p308.

101. Three journals have also contained a concentration of expert systems articles since 1985: Journal of Chemical Information and Computer Science (American Chemical Society), Chemometrics and Intelligent Laboratory Systems (Elsevier Science Publishers), and TRAC, Trends in Analytical Chemistry (Elsevier Science Publishers).

102. Lindsay, R.K.; Buchanan, B.G.; Feigenbaum, E.A.; Lederberg, J.; "Applications of Artificial Intelligence for Chemical Inference: The DENDRAL Project", McGraw-Hill, New York, **1980**.

103. Deming, S.N.; Palasota, J.M.; Lee, J.; Sun, L.; "Computer-Assisted Optimization in High-Performance Liquid Chromatographic Method Development" *J. Chrom.* **1989**, 485, 15-25.

104. Duda, R.O.; Gaschnig, J.G.; "Knowledge-Based Expert Systems Come of Age", *Byte*, **September 1981**, p238.

105. Tello,E.; "The Languages of AI Research", *PC Magazine*, April 16, **1985**, p173.

106. Harmon,P.; King,D.; "Expert Systems - Artificial Intelligence In Business", John Wiley and Sons Inc., New York, New York, **1985**.

107. Lee, N.S.; Roach, J.W. "GUESS/1: A General Purpose Expert System Shell", Thesis, Department of Computer Science, Virginia Polytechnic Institute and State University, **1985**.

108. Wipke, W.T.; Dyott, T.M. *JACS* **1974**, 96, 15, 4834-4842.

109. Molecular Design Limited MACCS Revision 5.0 Reference Manual, Appendix B.

110. The author wishes to thank Dr. Robert C. Klute for his gracious assistance.

111. "1989 Application Guide for chiral column selection" Daicel Chemical Industry, Chiral Technologies Inc., Exton, PA, 19341, (215) 594-2100. Sections Reprinted with Permission.

112. A factor of 0.8 is suggested by reference 80 when grid values are either 2 or 4. However calculations throughout that reference appear to be based on a grid factor of 0.5, which was adopted for use in this work.

113. A table of the scale factors can be found in Reference 80 page 279. The value actually used is (S * 1000). See "scale_factor (trait_number)" in module NEW.C in Appendix C.

114. Snyder, L.R.; Personal Communication.

115. Atkins, P.W. "Physical Chemistry, 3rd Edition" W.H. Freeman and Company, New York, **1986**, p579.

116. SPARC's batch mode generates polarizability values placing the decimal place one place to the left. Values reported were checked against SPARC's menu driven system. CHIRULE accounts for this difference in its normalization factor.

117. Atkins, P.W. "Physical Chemistry, 4th Edition" W.H. Freeman and Company, New York, **1990**, pp959-960.

118. Castellan, Gilbert W. "Physical Chemistry, 2nd Edition" Addison-Wesley, Reading, MA, **1971**, p624 ff.

119. Berry, R. Stephen; Rice, Stuart A.; Ross, John "Physical Chemistry" John Wiley & Sons, New York, **1980**, p411.

120. Lowry, Thomas H.; Richardson, Kathleen S. "Mechanism and Theory in Organic Chemistry, 2nd Ed" Harper and Row, New York **1981** p163.

121. Moelwyn-Hughes, E.A. "Physical Chemistry, Second Revised Edition" Pergamon Press, New York, **1961/1978**, p383.

122. Kamlet-Taft H-bonding constants provided by Dr. Taft.

123. Handbook of Chemistry and Physics, 60th Edition, Robert C. Weast, ed.; CRC Press, Inc.: Boca Raton, Florida, **1980**, p. E-27ff.

124. Wipke, W.T.; Krishnan, S.K.; Ouchi, G.I.; "Hash Functions for Rapid Storage and Retrieval of Chemical Structures" *JCICS* **1978**, 18, 32-37.

125. Pine, S.H.; Hendrickson, J.B.; Cram, J.D.; Hammond, G.S. "Organic Chemistry", 4th Ed., McGraw-Hill, New York. **1980**, p13.

126. The grant, written by the author and received in 1990, covered almost $80,000 worth of computer and networking equipment from Intel's Academic Relations group. It provided not only this equipment and additional equipment for use in the Laboratory Automation and Instrument Design groups ACS short course, but also the equipment required to establish a departmental network.

127. DOS 5.0 uses the upper memory block, the first 64Kb of memory after 1024K, to load itself into at bootup. DESQview normally uses this area to load itself into. With DOS 5.0 using this area with the DOS 5.0 "DOS=HIGH" feature, DESQview uses regular memory, eliminating the extra memory gained. In fact, the memory available to any DESQview window is actually 15K less with DOS 5.0 compared with DOS 4.01.

128. Alchemy from Tripos Associates, 1699 South Hanley, Suite 303, Saint Louis, MO 63144-2913, (314) 647-1099.

129. "VPI Prolog User Manual" John Deighan, Dr. John Roach, Department of Computer Science, Virginia Tech, Blacksburg, VA 26061.

130. FTP Software, 2 High Street, North Andover, MA 01845-2620, (508) 685-3300.

EXPERT SYSTEM SHELLS IN CHEMISTRY: CHIRULE - A CHIRAL CHROMATOGRAPHIC COLUMN SELECTION SYSTEM USING SIMILARITY SEARCHING AND PERSONAL CONSTRUCT THEORY

by

Scott T. Stauffer

APPENDICES

APPENDIX A: Dialog of a Session Using KEY . . . . . Page 341
APPENDIX B: OPTIMER Rulebase Generated by KEY . . . Page 352
APPENDIX C: KEY Source Code, Microsoft C . . . . . Page 354
APPENDIX D: KEY File Formats (ELM, TRT, GRD, RB) . Page 386
APPENDIX E: KEY Inference Engine Code, Prolog . . . Page 388
APPENDIX F: CHESS Source Code, Prolog . . . . . Page 395
APPENDIX G: Knowledge Bases Used by CHESS . . . . . Page 415
APPENDIX H: OPTIMER Testing Results . . . . . Page 425
APPENDIX I: CHIRULE Testing Results . . . . . Page 442
APPENDIX J: Daicel Chiral Column Selection Guide . Page 493

The appendices to the dissertation are contained in a separate volume. They consist of program listings for CHIRULE and CHESS, file formats for KEY, and dialog from tests that are summarized within the dissertation.

VITA

Scott Timothy Stauffer was born in Lancaster, Pennsylvania, and graduated from Donegal High School in Mount Joy, Pennsylvania. In the fall of 1979, he entered Elizabethtown College in Elizabethtown, Pennsylvania majoring in Chemistry. There he became involved with the Computer Science Department that had just begun a degree program, and graduated with a Bachelor of Science degree majoring in both Chemistry and Computer Science.

In the fall of 1984, he entered the Chemistry Ph.D. program at Virginia Tech. Intel Corporation's Components Research invited him to automate their R1 research fabrication facility in the summer of 1989. Upon returning from that experience, the Chemistry Department asked him to design and install a document preparation system for their main office, as well as the departmental network on which it would depend.

In June, 1993, he accepted full time employment at Lancaster Laboratories Inc. in Lancaster, Pennsylvania.

EXPERT SYSTEM SHELLS IN CHEMISTRY: CHIRULE - A CHIRAL CHROMATOGRAPHIC COLUMN SELECTION SYSTEM USING SIMILARITY SEARCHING AND PERSONAL CONSTRUCT THEORY

by

Scott T. Stauffer

VOLUME II - APPENDICES

APPENDIX A: Dialog of a Session Using KEY . . . . . Page 341
APPENDIX B: OPTIMER Rulebase Generated by KEY . . . Page 352
APPENDIX C: KEY Source Code, Microsoft C . . . . . Page 354
APPENDIX D: KEY File Formats (ELM, TRT, GRD, RB) . Page 386
APPENDIX E: KEY Inference Engine Code, Prolog . . . Page 388
APPENDIX F: CHESS Source Code, Prolog . . . . . Page 395
APPENDIX G: Knowledge Bases Used by CHESS . . . . . Page 415
APPENDIX H: OPTIMER Testing Results . . . . . Page 425
APPENDIX I: CHIRULE Testing Results . . . . . Page 442
APPENDIX J: Daicel Chiral Column Selection Guide . Page 493

Each of the appendices presented in this volume are preceded by a brief description of its contents. The descriptions contain references to the specific sections within the dissertation where the appendices are explained in greater detail.

APPENDIX A: Dialog of a Session Using KEY

Appendix A contains the complete transcript of the session where OPTIMER was developed. Section IV.B describes the various stages of development undergone when using KEY. The transcript begins with development of the knowledge grid, then demonstrates analyzing the grid, saving the rulebase, and running the rulebase with several examples. A **bold** font is used to display the information provided to KEY by the user.

```
B>key6

Below are listed the display modes that are currently available for use.  The
     numbers indicate columns(or characters) by rows(or lines) currently
     being shown by this display:

                    1)  80 x 25 - Any Display

                    2)  80 x 43 - Enhanced Display

                    3) 132 x 25 - Multi Display

                    4) 132 x 43 - Multi Display

                    5)  80 x 60 - Multi Display

                    6) 120 x 25 - Enhanced Display

                    7) 120 x 43 - Enhanced Display

                    8) 132 x 25 - Color Display

Please enter the number corresponding to the current display size: 2

Your display should display 80 columns and 43 rows.


Please enter the names of the elements.
    Your choices for obtaining the element names are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved element list
    ENTER - enter a new element list
    EDIT - edit the list of elements
    DEL - delete the list of elements
    SAVE - save the list of elements
    LIST - list out the elements
    DONE - finished entering element list
    HELP - print out this list of commands


Please enter your choice here -> enter

Now enter the names of the elements, one to a line, or the name of a command.

There are 3 commands you can enter, and they are:
    LIST - list out the elements
    DONE - finished entering element list
    HELP - print out this list of commands


Name of element 1 -> p 4
     Three letter abbreviation for element : p-4 -> 4


Name of element 2 -> p 5-7
     Three letter abbreviation for element : p-5-7 -> 5-7

Name of element 3 -> p 8-9
     Three letter abbreviation for element : p-8-9 -> 8-9

Name of element 4 -> p 10-11
     Three letter abbreviation for element : p-10-11 -> 10-11

Name of element 5 -> p 12
     Three letter abbreviation for element : p-12 -> 12

Name of element 6 -> p 13
     Three letter abbreviation for element : p-13 -> 13

Name of element 7 -> p 14
     Three letter abbreviation for element : p-14 -> 14

Name of element 8 -> p 15
     Three letter abbreviation for element : p-15 -> 15

Name of element 9 -> p 16
     Three letter abbreviation for element : p-16 -> 16

Name of element 10 -> done


    Your choices for obtaining the element names are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved element list
    ENTER - enter a new element list
    EDIT - edit the list of elements
    DEL - delete the list of elements
    SAVE - save the list of elements
    LIST - list out the elements
    DONE - finished entering element list
    HELP - print out this list of commands

Please enter your choice here -> list


There are 9 elements currently.  Their names follow :

The name of element 1 is (p-4),
                    and its abbreviation is (4).
The name of element 2 is (p-5-7),
                    and its abbreviation is (5-7).
The name of element 3 is (p-8-9),
                    and its abbreviation is (8-9).
The name of element 4 is (p-10-11),
                    and its abbreviation is (10-).
The name of element 5 is (p-12),
                    and its abbreviation is (12).
The name of element 6 is (p-13),
                    and its abbreviation is (13).
The name of element 7 is (p-14),
                    and its abbreviation is (14).
The name of element 8 is (p-15),
                    and its abbreviation is (15).
The name of element 9 is (p-16),
                    and its abbreviation is (16).

Please hit RETURN to continue -


    Your choices for obtaining the element names are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved element list
    ENTER - enter a new element list
    EDIT - edit the list of elements
    DEL - delete the list of elements
    SAVE - save the list of elements
    LIST - list out the elements
    DONE - finished entering element list
    HELP - print out this list of commands

Please enter your choice here -> save

Please enter name of the file to contain the saved element list.

If no extension is entered, .ELM will be assumed. (type RETURN to exit)
    Filename -> chiral
```

```
    Is CHIRAL.ELM the correct file name? (*YES or NO) ->

The element list has been saved.


    Your choices for obtaining the element names are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved element list
    ENTER - enter a new element list
    EDIT - edit the list of elements
    DEL - delete the list of elements
    SAVE - save the list of elements
    LIST - list out the elements
    DONE - finished entering element list
    HELP - print out this list of commands

Please enter your choice here -> done


Next, enter the names of the traits.
    Your choices for obtaining the trait names are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved trait list
    ENTER - enter a new trait list
    EDIT - edit the list of traits
    DEL - delete the list of traits
    SAVE - save the list of traits
    LIST - list out the traits
    DONE - finished entering trait list
    HELP - print out this list of commands

Please enter your choice here -> enter

Now enter the names of the traits, one to a line, or the name of a command.

There are 4 commands you can enter, and they are:
    LIST - list out the traits
    NEXT - go on to the next triad of traits
    DONE - finished entering trait list
    HELP - print out this list of commands

Consider the following three elements :
                    p-4
                  p-10-11
                    p-12

Please determine a trait that two of the above elements have that the third
     does not, and enter the name of that trait below.

Name of trait 1 -> CdO
     Now please enter the opposite of CdO here -> no CdO
     Think of a name that can be used to describe this trait, and
        enter it here -> carbonyl


Consider the following three elements :
                    p-12
                    p-15
                   p-5-7

Please determine a trait that two of the above elements have that the third
     does not, and enter the name of that trait below.

Name of trait 2 -> has benz
     Now please enter the opposite of has-benz here -> no benz
     Think of a name that can be used to describe this trait, and
        enter it here -> aromatic ring


Consider the following three elements :
                    p-4
                  p-10-11
                    p-12

Please determine a trait that two of the above elements have that the third
     does not, and enter the name of that trait below.

Name of trait 3 -> help


There are 4 commands you can enter, and they are:
    LIST - list out the traits
    NEXT - go on to the next triad of traits
    DONE - finished entering trait list
    HELP - print out this list of commands


Consider the following three elements :
                    p-4
                  p-10-11
                    p-12

Please determine a trait that two of the above elements have that the third
     does not, and enter the name of that trait below.

Name of trait 3 -> next


Consider the following three elements :
                    p-4
                   p-8-9
                   p-5-7

Please determine a trait that two of the above elements have that the third
     does not, and enter the name of that trait below.

Name of trait 3 -> OsH
     Now please enter the opposite of OsH here -> no OsH
     Think of a name that can be used to describe this trait, and
        enter it here -> hydroxyl


Consider the following three elements :
                    p-14
                    p-15
                     p-4

Please determine a trait that two of the above elements have that the third
     does not, and enter the name of that trait below.

Name of trait 4 -> done


    Your choices for obtaining the trait names are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved trait list
    ENTER - enter a new trait list
    EDIT - edit the list of traits
    DEL - delete the list of traits
    SAVE - save the list of traits
    LIST - list out the traits
    DONE - finished entering trait list
    HELP - print out this list of commands

Please enter your choice here -> edit
```

```
Your choices for editing the trait list are as follows:

There are 7 commands you can enter, and they are:
     CHANGE - change one trait in the list of traits
     ADD - add one trait to the list of traits
     REMOVE - remove one trait from the list of traits
     LIST - list out the traits
     LIST-G - list out the grid
     DONE - finished entering trait list
     HELP - print out this list of commands

Please enter your choice here -> list

There are 3 traits currently.  Their names follow :

The name of trait 1 is {CdO}, its opposite is {no-CdO},
    and a description of this trait is {carbonyl}.
The name of trait 2 is {has-benz}, its opposite is {no-benz},
    and a description of this trait is {aromatic-ring}.
The name of trait 3 is {OsH}, its opposite is {no-OsH},
    and a description of this trait is {hydroxyl}.

Please hit RETURN to continue -


Your choices for editing the trait list are as follows:

There are 7 commands you can enter, and they are:
     CHANGE - change one trait in the list of traits
     ADD - add one trait to the list of traits
     REMOVE - remove one trait from the list of traits
     LIST - list out the traits
     LIST-G - list out the grid
     DONE - finished entering trait list
     HELP - print out this list of commands

Please enter your choice here -> add


You want to add another trait BEFORE what trait number (enter '0' to
     LIST traits, any negative number to EXIT without additions,
     or any large number to append the trait to the end) -> 8

The name of trait 4 is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) N3
The trait's opposite is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) no N3
The description of trait 4 is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) amine
               The name of trait 4 is {N3},
                    and its opposite is {no-N3}.
               The description of trait 4 is {amine},


Your choices for editing the trait list are as follows:

There are 7 commands you can enter, and they are:
     CHANGE - change one trait in the list of traits
     ADD - add one trait to the list of traits
     REMOVE - remove one trait from the list of traits
     LIST - list out the traits
     LIST-G - list out the grid
     DONE - finished entering trait list
     HELP - print out this list of commands

Please enter your choice here -> add

You want to add another trait BEFORE what trait number (enter '0' to
     LIST traits, any negative number to EXIT without additions,
     or any large number to append the trait to the end) -> 7


The name of trait 5 is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) NP
The trait's opposite is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) no NP
The description of trait 5 is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) primary amine
               The name of trait 5 is {NP},
                    and its opposite is {no-NP}.
               The description of trait 5 is {primary-amine},


Your choices for editing the trait list are as follows:

There are 7 commands you can enter, and they are:
     CHANGE - change one trait in the list of traits
     ADD - add one trait to the list of traits
     REMOVE - remove one trait from the list of traits
     LIST - list out the traits
     LIST-G - list out the grid
     DONE - finished entering trait list
     HELP - print out this list of commands

Please enter your choice here -> add


You want to add another trait BEFORE what trait number (enter '0' to
     LIST traits, any negative number to EXIT without additions,
     or any large number to append the trait to the end) -> 6

The name of trait 6 is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) COOH
The trait's opposite is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) no COOH
The description of trait 6 is currently {}.  What would you
     like it to be? (RETURN to remain unchanged) carboxylic acid
               The name of trait 6 is {COOH},
                    and its opposite is {no-COOH}.
               The description of trait 6 is {carboxylic-acid},


Your choices for editing the trait list are as follows:

There are 7 commands you can enter, and they are:
     CHANGE - change one trait in the list of traits
     ADD - add one trait to the list of traits
     REMOVE - remove one trait from the list of traits
     LIST - list out the traits
     LIST-G - list out the grid
     DONE - finished entering trait list
     HELP - print out this list of commands


Please enter your choice here -> list

There are 6 traits currently.  Their names follow :

The name of trait 1 is {CdO}, its opposite is {no-CdO},
    and a description of this trait is {carbonyl}.
The name of trait 2 is {has-benz}, its opposite is {no-benz},
    and a description of this trait is {aromatic-ring}.
The name of trait 3 is {OsH}, its opposite is {no-OsH},
    and a description of this trait is {hydroxyl}.
The name of trait 4 is {N3}, its opposite is {no-N3},
    and a description of this trait is {amine}.
The name of trait 5 is {NP}, its opposite is {no-NP},
    and a description of this trait is {primary-amine}.
```

```
The name of trait 6 is {COOH}, its opposite is {no-COOH},
    and a description of this trait is {carboxylic-acid}.

Please hit RETURN to continue -


Your choices for editing the trait list are as follows:

There are 7 commands you can enter, and they are:
    CHANGE - change one trait in the list of traits
    ADD - add one trait to the list of traits
    REMOVE - remove one trait from the list of traits
    LIST - list out the traits
    LIST-G - list out the grid
    DONE - finished entering trait list
    HELP - print out this list of commands

Please enter your choice here -> done


    Your choices for obtaining the trait names are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved trait list
    ENTER - enter a new trait list
    EDIT - edit the list of traits
    DEL - delete the list of traits
    SAVE - save the list of traits
    LIST - list out the traits
    DONE - finished entering trait list
    HELP - print out this list of commands

Please enter your choice here -> save

Please enter name of the file to contain the saved trait list.
If no extension is entered, .TRT will be assumed. (type RETURN to exit)
    Filename -> chiral
    Is CHIRAL.TRT the correct file name? (*YES or NO) ->
The trait list has been saved.


    Your choices for obtaining the trait names are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved trait list
    ENTER - enter a new trait list
    EDIT - edit the list of traits
    DEL - delete the list of traits
    SAVE - save the list of traits
    LIST - list out the traits
    DONE - finished entering trait list
    HELP - print out this list of commands

Please enter your choice here -> done


Now enter each element's value for every trait.
    Your choices for obtaining the element's trait values are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved value list
    ENTER - enter a new value list
    EDIT - edit the list of values
    DEL - delete the list of values
    SAVE - save the list of values
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> enter


Please rate each of the following elements on a scale of 1 to 5, where
    {CdO} is 5 and {no-CdO} is 1.  (type any negative
    number to exit, or '0' to remain unchanged)

Value for p-4 is currently 0, new value is -> 5
Value for p-5-7 is currently 0, new value is -> 5
Value for p-8-9 is currently 0, new value is -> 5
Value for p-10-11 is currently 0, new value is -> 1
Value for p-12 is currently 0, new value is -> 1
Value for p-13 is currently 0, new value is -> 1
Value for p-14 is currently 0, new value is -> 3
Value for p-15 is currently 0, new value is -> 3
Value for p-16 is currently 0, new value is -> 3


Please rate each of the following elements on a scale of 1 to 5, where
    {has-benz} is 5 and {no-benz} is 1.  (type any negative
    number to exit, or '0' to remain unchanged)

Value for p-4 is currently 0, new value is -> 5
Value for p-5-7 is currently 0, new value is -> 5
Value for p-8-9 is currently 0, new value is -> 5
Value for p-10-11 is currently 0, new value is -> 5
Value for p-12 is currently 0, new value is -> 5
Value for p-13 is currently 0, new value is -> 5
Value for p-14 is currently 0, new value is -> 1
Value for p-15 is currently 0, new value is -> 1
Value for p-16 is currently 0, new value is -> 1


Please rate each of the following elements on a scale of 1 to 5, where
    {OsH} is 5 and {no-OsH} is 1.  (type any negative
    number to exit, or '0' to remain unchanged)

Value for p-4 is currently 0, new value is -> 5
Value for p-5-7 is currently 0, new value is -> 1
Value for p-8-9 is currently 0, new value is -> 1
Value for p-10-11 is currently 0, new value is -> 5
Value for p-12 is currently 0, new value is -> 1
Value for p-13 is currently 0, new value is -> 1
Value for p-14 is currently 0, new value is -> 3
Value for p-15 is currently 0, new value is -> 3
Value for p-16 is currently 0, new value is -> 3


Please rate each of the following elements on a scale of 1 to 5, where
    {N3} is 5 and {no-N3} is 1.  (type any negative
    number to exit, or '0' to remain unchanged)

Value for p-4 is currently 0, new value is -> 3
Value for p-5-7 is currently 0, new value is -> 5
Value for p-8-9 is currently 0, new value is -> 1
Value for p-10-11 is currently 0, new value is -> 3
Value for p-12 is currently 0, new value is -> 5
Value for p-13 is currently 0, new value is -> 1
Value for p-14 is currently 0, new value is -> 3
Value for p-15 is currently 0, new value is -> 3
Value for p-16 is currently 0, new value is -> 3


Please rate each of the following elements on a scale of 1 to 5, where
    {NP} is 5 and {no-NP} is 1.  (type any negative
    number to exit, or '0' to remain unchanged)

Value for p-4 is currently 0, new value is -> 3
Value for p-5-7 is currently 0, new value is -> 3
```

```
Value for p-8-9 is currently 0, new value is -> 3
Value for p-10-11 is currently 0, new value is -> 3
Value for p-12 is currently 0, new value is -> 3
Value for p-13 is currently 0, new value is -> 3
Value for p-14 is currently 0, new value is -> 5
Value for p-15 is currently 0, new value is -> 1
Value for p-16 is currently 0, new value is -> 1


Please rate each of the following elements on a scale of 1 to 5, where
     (COOH) is 5 and (no-COOH) is 1.  (type any negative
     number to exit, or '0' to remain unchanged)

Value for p-4 is currently 0, new value is -> 3
Value for p-5-7 is currently 0, new value is -> 3
Value for p-8-9 is currently 0, new value is -> 3
Value for p-10-11 is currently 0, new value is -> 3
Value for p-12 is currently 0, new value is -> 3
Value for p-13 is currently 0, new value is -> 3
Value for p-14 is currently 0, new value is -> 3
Value for p-15 is currently 0, new value is -> 5
Value for p-16 is currently 0, new value is -> 1


    Your choices for obtaining the element's trait values are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved value list
    ENTER - enter a new value list
    EDIT - edit the list of values
    DEL - delete the list of values
    SAVE - save the list of values
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> list

   4 = p-4
     5-7 = p-5-7
         8-9 = p-8-9
             10- = p-10-11
                 12 = p-12
                     13 = p-13
                         14 = p-14
                             15 = p-15
                                 16 = p-16
  4 5-7 8-9 10-  12  13  14  15  16     trait pairs
5   5   5   1   1   1   3   3   3 CdO = 5, no-CdO = 1
5   5   5   5   5   5   1   1   1 has-benz = 5, no-benz = 1
5   1   1   5   1   1   3   3   3 OsH = 5, no-OsH = 1
3   5   1   3   5   1   3   3   3 N3 = 5, no-N3 = 1
3   3   3   3   3   3   5   1   1 NP = 5, no-NP = 1
3   3   3   3   3   3   3   5   1 COOH = 5, no-COOH = 1

Please hit RETURN to continue -


    Your choices for obtaining the element's trait values are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved value list
    ENTER - enter a new value list
    EDIT - edit the list of values
    DEL - delete the list of values
    SAVE - save the list of values
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> save

Please enter name of the file to contain the saved trait list.
    Is CHIRAL.GRD the correct file name? (*YES or NO) ->
The trait list has been saved.


    Your choices for obtaining the element's trait values are as follows:

There are 8 commands you can enter, and they are:
    GET - retrieve a saved value list
    ENTER - enter a new value list
    EDIT - edit the list of values
    DEL - delete the list of values
    SAVE - save the list of values
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> done


Your choices for analyzing the present grid are as follows:

There are 10 commands you can enter, and they are:
    CH-MATCH - change the value at which matches are detected
    E-ANAL - analyze the elements for similarities
    T-ANAL - analyze the traits for similarities
    RATE-T - determine the importance of each trait
    RULES - generate rules from the current grid
    AUTO - automatically do the analysis and rule generation
    SAVE-ALL - save the elements, traits, and grid.
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> auto

     The procedure for automated analysis is about to begin.  This procedure
will take you through the analysis of both the element and traits to resolve
many inconsistencies found to be present.  Following this procedure, options
will be presented to allow generation of a rule-based system reflecting
the information contained in the grid.


Please hit RETURN to continue -


Below are listed the cluster analysis results of the elements:

      element            1   2   3   4   5   6   7   8   9
 ================================================================
  1.              p-4   --  75  75  84  59  59  59  50  50
  2.            p-5-7       --  84  59  84  67  50  42  42
  3.            p-8-9           --  59  67  84  50  42  42
  4.          p-10-11               --  75  75  59  50  50
  5.            p-12                    --  84  50  42  42
  6.            p-13                        --  50  42  42
  7.            p-14                            --  75  75
  8.            p-15                                --  84
  9.            p-16                                    --

Please hit RETURN to continue -

Below are listed the cluster analysis results of the traits:
  (Negative values indicate that traits are opposites.)

        trait            1   2   3   4   5   6
 ================================================
```

```
1.              CdO    --    0   34   34    0   12
2.        has-benz   -66   --  -22    0   12    0
3.             OsH    12    0   --   34    0   12
4.              N3    12    0  -11   --   23   34
5.              NP    23  -11   23   23   --   67
6.            COOH    34    0   12   56   45   --

Note the numbers in the lower left-hand side of the table result from
  inverting one of the trait poles before doing the analysis.

Please hit RETURN to continue -

Please rate each of the following traits based on their importance to the
  problem at hand, using a scale of 1 (least important) to 5 (most important).
  (type any negative number to exit, or '0' to remain unchanged)

The importance of trait {carbonyl} is currently 0, new value is -> 5
The importance of trait {aromatic-ring} is currently 0, new value is -> 5
The importance of trait {hydroxyl} is currently 0, new value is -> 5
The importance of trait {amine} is currently 0, new value is -> 5
The importance of trait {primary-amine} is currently 0, new value is -> 5
The importance of trait {carboxylic-acid} is currently 0, new value is -> 5


Your choices for generating rules from the present grid are as follows:

There are 6 commands you can enter, and they are:
    T-PROLOG - generate a turbo-prolog system
    HC-PROLOG - generate an hc-prolog system
    PROLOG - generate an prolog system with entailments
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> list

   4 = p-4
     5-7 = p-5-7
         8-9 = p-8-9
             10- = p-10-11
                 12 = p-12
                     13 = p-13
                         14 = p-14
                             15 = p-15
                                 16 = p-16
   4 5-7 8-9 10-  12  13  14  15  16 (rating)   trait pairs
5   5   5   1   1   1   3   3   3     (5)   CdO = 5, no-CdO = 1
5   5   5   5   5   5   1   1   1     (5)   has-benz = 5, no-benz = 1
5   1   1   5   1   1   3   3   3     (5)   OsH = 5, no-OsH = 1
3   5   1   3   5   1   3   3   3     (5)   N3 = 5, no-N3 = 1
3   3   3   3   3   3   5   1   1     (5)   NP = 5, no-NP = 1
3   3   3   3   3   3   3   5   1     (5)   COOH = 5, no-COOH = 1

Please hit RETURN to continue -


Your choices for generating rules from the present grid are as follows:

There are 6 commands you can enter, and they are:
    T-PROLOG - generate a turbo-prolog system
    HC-PROLOG - generate an hc-prolog system
    PROLOG - generate an prolog system with entailments
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> hc-prolog

Please enter the file name for the rule base.
    is CHIRAL.HC the correct file name? (*YES or NO) ->
Rule generation for HC Prolog is about to begin.  Note that since there are
    9 elements and 6 traits, over 108 rules could be generated.
Would you like to proceed?   (*YES or NO) ->

What is a short description of the area you are investigating?
                                    (enter it here) -> choose a chiral column
In looking at choose-a-chiral, what is a description of the final
    answer you are looking for?     (enter it here) -> a column
In classifying the different a-columns, what is a description
    of the traits you are using?    (enter it here) -> functional groups
If all of the functional-grous deal with one particular item, what
    is a description of that item? (Return for none) -> molecule

The rule list has been saved in the file CHIRAL.HC.


Your choices for generating rules from the present grid are as follows:

There are 6 commands you can enter, and they are:
    T-PROLOG - generate a turbo-prolog system
    HC-PROLOG - generate an hc-prolog system
    PROLOG - generate an prolog system with entailments
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> done


Your choices for analyzing the present grid are as follows:

There are 10 commands you can enter, and they are:
    CH-MATCH - change the value at which matches are detected
    E-ANAL - analyze the elements for similarities
    T-ANAL - analyze the traits for similarities
    RATE-T - determine the importance of each trait
    RULES - generate rules from the current grid
    AUTO - automatically do the analysis and rule generation
    SAVE-ALL - save the elements, traits, and grid.
    LIST - list out the values
    DONE - finished entering value list
    HELP - print out this list of commands

Please enter your choice here -> done

                The element list has been deleted.
                The trait list has been deleted.

B>
B>


B>type chiral.hc
;                       ******* File CHIRAL.HC *******
(assert
; generated using PCT

((question "chiral" "choose-a-chiral" "a-column" "functional-grou" "molecule" (
      (carbonyl CdO no-CdO)
      (aromatic-ring has-benz no-benz)
      (hydroxyl OsH no-OsH)
      (amine N3 no-N3)
      (primary-amine NP no-NP)
```

```
     (carboxylic-acid COOH no-COOH)
          )     ))

((maxrules 108))

((chiral 1 p-4 44) if (fact carbonyl CdO))
((chiral 2 p-4 -44) if (fact carbonyl no-CdO))
((chiral 3 p-5-7 44) if (fact carbonyl CdO))
((chiral 4 p-5-7 -44) if (fact carbonyl no-CdO))
((chiral 5 p-8-9 44) if (fact carbonyl CdO))
((chiral 6 p-8-9 -44) if (fact carbonyl no-CdO))
((chiral 7 p-10-11 -44) if (fact carbonyl CdO))
((chiral 8 p-10-11 44) if (fact carbonyl no-CdO))
((chiral 9 p-12 -44) if (fact carbonyl CdO))
((chiral 10 p-12 44) if (fact carbonyl no-CdO))
((chiral 11 p-13 -44) if (fact carbonyl CdO))
((chiral 12 p-13 44) if (fact carbonyl no-CdO))
((chiral 13 p-4 44) if (fact aromatic-ring has-benz))
((chiral 14 p-4 -44) if (fact aromatic-ring no-benz))
((chiral 15 p-5-7 44) if (fact aromatic-ring has-benz))
((chiral 16 p-5-7 -44) if (fact aromatic-ring no-benz))
((chiral 17 p-8-9 44) if (fact aromatic-ring has-benz))
((chiral 18 p-8-9 -44) if (fact aromatic-ring no-benz))
((chiral 19 p-10-11 44) if (fact aromatic-ring has-benz))
((chiral 20 p-10-11 -44) if (fact aromatic-ring no-benz))
((chiral 21 p-12 44) if (fact aromatic-ring has-benz))
((chiral 22 p-12 -44) if (fact aromatic-ring no-benz))
((chiral 23 p-13 44) if (fact aromatic-ring has-benz))
((chiral 24 p-13 -44) if (fact aromatic-ring no-benz))
((chiral 25 p-14 -44) if (fact aromatic-ring has-benz))
((chiral 26 p-14 44) if (fact aromatic-ring no-benz))
((chiral 27 p-15 -44) if (fact aromatic-ring has-benz))
((chiral 28 p-15 44) if (fact aromatic-ring no-benz))
((chiral 29 p-16 -44) if (fact aromatic-ring has-benz))
((chiral 30 p-16 44) if (fact aromatic-ring no-benz))
((chiral 31 p-4 44) if (fact hydroxyl OsH))
((chiral 32 p-4 -44) if (fact hydroxyl no-OsH))
((chiral 33 p-5-7 -44) if (fact hydroxyl OsH))
((chiral 34 p-5-7 44) if (fact hydroxyl no-OsH))
((chiral 35 p-8-9 -44) if (fact hydroxyl OsH))
((chiral 36 p-8-9 44) if (fact hydroxyl no-OsH))
((chiral 37 p-10-11 44) if (fact hydroxyl OsH))
((chiral 38 p-10-11 -44) if (fact hydroxyl no-OsH))
((chiral 39 p-12 -44) if (fact hydroxyl OsH))
((chiral 40 p-12 44) if (fact hydroxyl no-OsH))
((chiral 41 p-13 -44) if (fact hydroxyl OsH))
((chiral 42 p-13 44) if (fact hydroxyl no-OsH))
((chiral 43 p-5-7 44) if (fact amine N3))
((chiral 44 p-5-7 -44) if (fact amine no-N3))
((chiral 45 p-8-9 -44) if (fact amine N3))
((chiral 46 p-8-9 44) if (fact amine no-N3))
((chiral 47 p-12 44) if (fact amine N3))
((chiral 48 p-12 -44) if (fact amine no-N3))
((chiral 49 p-13 -44) if (fact amine N3))
((chiral 50 p-13 44) if (fact amine no-N3))
((chiral 51 p-14 44) if (fact primary-amine NP))
((chiral 52 p-14 -44) if (fact primary-amine no-NP))
((chiral 53 p-15 -44) if (fact primary-amine NP))
((chiral 54 p-15 44) if (fact primary-amine no-NP))
((chiral 55 p-16 -44) if (fact primary-amine NP))
((chiral 56 p-16 44) if (fact primary-amine no-NP))
((chiral 57 p-15 44) if (fact carboxylic-acid COOH))
((chiral 58 p-15 -44) if (fact carboxylic-acid no-COOH))
((chiral 59 p-16 -44) if (fact carboxylic-acid COOH))
((chiral 60 p-16 44) if (fact carboxylic-acid no-COOH))

)
;  *** END OF assert
```

```
$ hcc res
Report any bugs (including, hopefully, a short example program)
to delghan@csgrad

<Enter Carriage Return to continue>

Welcome to VPI Prolog, version 1.039, last compiled on June 18, 1992

To exit Prolog, enter: quit
For help, enter:
    help           - to get this display
    help topics    - to get a list of topics
                     for which help is available
    help prolog    - for help with invoking VPI Prolog,
                     including available command line options
    help general   - for a summary of valid input
    help syntax    - for a summary of VPI Prolog syntax
    help edinburgh - for a summary of Edinburgh syntax
    help builtins  - for a list of builtin predicates and functions
    help debug     - for help with debugging
   YES
   YES
   YES
   YES
   YES
   YES
   YES
?- consult key.hc
?-
?- load_all_db
   YES
?- run
What is the name of the rule base you want to use?
      Please enter it here -> chiral-b

The Rulebase ->chiral-b<- has been loaded.


In looking at choose-a-chiral, trying to determine a column,
     what molecule would you like to investigate?
     Enter the molecule name here --> MANDELIC ACID
  [MANDELIC ACID] currently exists in runtime Molecule database.....


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
       (molecule name is ->MANDELIC ACID<- )

   Looking for pre-exisiting CdO with item MANDELIC ACID.
   Found CdO predicate for carbonyl, will it succeed?
         Attempting CdO using MANDELIC ACID.....
Looking for a C=O in MANDELIC ACID:
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
*    Found C=O bond in MANDELIC ACID at atom 9.
         Executed CdO using MANDELIC ACID.
*   For carbonyl, Found that CdO exists. [MANDELIC ACID]

   Looking for pre-exisiting has-benz with item MANDELIC ACID.
   Looking for pre-exisiting has-benz, but having it fail.
   Looking for pre-exisiting opposite of has-benz.
WHICH OF THE FOLLOWING IS TRUE ABOUT    aromatic-ring
      A] has-benz
      B] no-benz
Please type in your response as A/B ->  A
*  You chose A, which was has-benz.

   Looking for pre-exisiting OsH with item MANDELIC ACID.
   Found OsH predicate for hydroxyl, will it succeed?
```

```
        Attempting OsH using MANDELIC ACID.....
Looking for a -OH in MANDELIC ACID:
  Found Oxygen at atom 8.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*     Found -OH bond in MANDELIC ACID at atom 8.
        Executed OsH using MANDELIC ACID.
*    For hydroxyl, Found that OsH exists. [MANDELIC ACID]

   Looking for pre-exisiting N3 with item MANDELIC ACID.
   Found N3 predicate for amine, will it succeed?
        Attempting N3 using MANDELIC ACID.....
Looking for an Amine in MANDELIC ACID:
   Looking for pre-exisiting N3, but having it fail.
     Found N3 predicate for amine, will it fail?
        Attempting N3 using MANDELIC ACID.....
Looking for an Amine in MANDELIC ACID:
*    For amine, Found N3 doesn't exist, so it's no-N3.

   Looking for pre-exisiting NP with item MANDELIC ACID.
   Found NP predicate for primary-amine, will it succeed?
        Attempting NP using MANDELIC ACID.....
Looking for a Primary Amine (NRH2) in MANDELIC ACID:
   Looking for pre-exisiting NP, but having it fail.
     Found NP predicate for primary-amine, will it fail?
        Attempting NP using MANDELIC ACID.....
Looking for a Primary Amine (NRH2) in MANDELIC ACID:
*    For primary-amine, Found NP doesn't exist, so it's no-NP.

   Looking for pre-exisiting COOH with item MANDELIC ACID.
   Found COOH predicate for carboxylic-acid, will it succeed?
        Attempting COOH using MANDELIC ACID.....
Looking for a Carboxylic acid (XC=OOH) in MANDELIC ACID:
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
  Have C-O at atom 11.  Is there a -OH bond?
*     Found Carboxylic Acid (COOH) in MANDELIC ACID at atom 9.
        Executed COOH using MANDELIC ACID.
*    For carboxylic-acid, Found that COOH exists. [MANDELIC ACID]

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
          (molecule name is ->MANDELIC ACID<- )
p-4    WITH A PROBABILITY OF    82.4384
p-8-9    WITH A PROBABILITY OF    74.7113
p-10-11    WITH A PROBABILITY OF    54.8416
p-15    WITH A PROBABILITY OF    54.8416
p-13    WITH A PROBABILITY OF    34.9719
p-5-7    WITH A PROBABILITY OF    34.9719
p-16    WITH A PROBABILITY OF    -16.1216
p-12    WITH A PROBABILITY OF    -53.0281
p-14    WITH A PROBABILITY OF    -68.64

The following is the data for (choose-a-chiral : A A A B B A).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose COOH.
   YES
?-
?- run
What is the name of the rule base you want to use?
      Please enter it here -> chiral-b

The Rulebase ->chiral-b<- is already loaded.


In looking at choose-a-chiral, trying to determine a-column,
     what molecule would you like to investigate?
     Enter the molecule name here --> MANDELAMIDE
  [MANDELAMIDE] currently exists in runtime Molecule database.....


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
       (molecule name is ->MANDELAMIDE<- )

   Looking for pre-exisiting CdO with item MANDELAMIDE.
   Found CdO predicate for carbonyl, will it succeed?
        Attempting CdO using MANDELAMIDE.....
Looking for a C=O in MANDELAMIDE:
  Found Oxygen at atom 8.  Is it a C-O?
  Found Oxygen at atom 10.  Is it a C=O?
*     Found C=O bond in MANDELAMIDE at atom 9.
        Executed CdO using MANDELAMIDE.
*    For carbonyl, Found that CdO exists. [MANDELAMIDE]

   Looking for pre-exisiting has-benz with item MANDELAMIDE.
   Looking for pre-exisiting has-benz, but having it fail.
   Looking for pre-exisiting opposite of has-benz.
WHICH OF THE FOLLOWING IS TRUE ABOUT     aromatic-ring
      A] has-benz
      B] no-benz
Please type in your response as A/B ->  A
*   You chose A, which was has-benz.

   Looking for pre-exisiting OsH with item MANDELAMIDE.
   Found OsH predicate for hydroxyl, will it succeed?
        Attempting OsH using MANDELAMIDE.....
Looking for a -OH in MANDELAMIDE:
  Found Oxygen at atom 8.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*     Found -OH bond in MANDELAMIDE at atom 8.
        Executed OsH using MANDELAMIDE.
*    For hydroxyl, Found that OsH exists. [MANDELAMIDE]

   Looking for pre-exisiting N3 with item MANDELAMIDE.
   Found N3 predicate for amine, will it succeed?
        Attempting N3 using MANDELAMIDE.....
Looking for an Amine in MANDELAMIDE:
*     Found Amine (-NR2) in MANDELAMIDE at atom 11.
        Executed N3 using MANDELAMIDE.
*    For amine, Found that N3 exists. [MANDELAMIDE]

   Looking for pre-exisiting NP with item MANDELAMIDE.
   Found NP predicate for primary-amine, will it succeed?
        Attempting NP using MANDELAMIDE.....
Looking for a Primary Amine (NRH2) in MANDELAMIDE:
*     Found Primary Amine (-NH2) in MANDELAMIDE at atom 11.
        Executed NP using MANDELAMIDE.
*    For primary-amine, Found that NP exists. [MANDELAMIDE]

   Looking for pre-exisiting COOH with item MANDELAMIDE.
   Found COOH predicate for carboxylic-acid, will it succeed?
        Attempting COOH using MANDELAMIDE.....
Looking for a Carboxylic acid (XC=OOH) in MANDELAMIDE:
  Found Oxygen at atom 8.  Is it a C-O?
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
   Looking for pre-exisiting COOH, but having it fail.
     Found COOH predicate for carboxylic-acid, will it fail?
        Attempting COOH using MANDELAMIDE.....
Looking for a Carboxylic acid (XC=OOH) in MANDELAMIDE:
```

```
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
*   For carboxylic-acid, Found COOH doesn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
                (molecule name is ->MANDELAMIDE<- )
p-4   WITH A PROBABILITY OF   82.4384
p-5-7   WITH A PROBABILITY OF   74.7113
p-10-11   WITH A PROBABILITY OF   54.8416
p-12   WITH A PROBABILITY OF   34.9719
p-8-9   WITH A PROBABILITY OF   34.9719
p-14   WITH A PROBABILITY OF   19.36
p-16   WITH A PROBABILITY OF   5.5616
p-13   WITH A PROBABILITY OF   -53.0281
p-15   WITH A PROBABILITY OF   -82.4384

The following is the data for (choose-a-chiral : A A A A A B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose no-COOH.
   YES
?-
?- run
What is the name of the rule base you want to use?
      Please enter it here -> chiral-b

The Rulebase ->chiral-b<- is already loaded.


In looking at choose-a-chiral, trying to determine a-column,
     what molecule would you like to investigate?
     Enter the molecule name here --> 1-PHENYLETHANOL
  [1-PHENYLETHANOL] currently exists in runtime Molecule database.....


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
        (molecule name is ->1-PHENYLETHANOL<- )

   Looking for pre-exisiting CdO with item 1-PHENYLETHANOL.
   Found CdO predicate for carbonyl, will it succeed?
         Attempting CdO using 1-PHENYLETHANOL.....
Looking for a C=O in 1-PHENYLETHANOL:
  Found Oxygen at atom 9.  Is it a C=O?
   Looking for pre-exisiting CdO, but having it fail.
     Found CdO predicate for carbonyl, will it fail?
         Attempting CdO using 1-PHENYLETHANOL.....
Looking for a C=O in 1-PHENYLETHANOL:
  Found Oxygen at atom 9.  Is it a C=O?
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

   Looking for pre-exisiting has-benz with item 1-PHENYLETHANOL.
   Looking for pre-exisiting has-benz, but having it fail.
   Looking for pre-exisiting opposite of has-benz.
WHICH OF THE FOLLOWING IS TRUE ABOUT    aromatic-ring
      A] has-benz
      B] no-benz
Please type in your response as A/B ->  A
*  You chose A, which was has-benz.

   Looking for pre-exisiting OsH with item 1-PHENYLETHANOL.
   Found OsH predicate for hydroxyl, will it succeed?
         Attempting OsH using 1-PHENYLETHANOL.....
Looking for a -OH in 1-PHENYLETHANOL:
  Found Oxygen at atom 9.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*     Found -OH bond in 1-PHENYLETHANOL at atom 9.
         Executed OsH using 1-PHENYLETHANOL.
*   For hydroxyl, Found that OsH exists. [1-PHENYLETHANOL]

   Looking for pre-exisiting N3 with item 1-PHENYLETHANOL.
   Found N3 predicate for amine, will it succeed?
         Attempting N3 using 1-PHENYLETHANOL.....
Looking for an Amine in 1-PHENYLETHANOL:
   Looking for pre-exisiting N3, but having it fail.
     Found N3 predicate for amine, will it fail?
         Attempting N3 using 1-PHENYLETHANOL.....
Looking for an Amine in 1-PHENYLETHANOL:
*   For amine, Found N3 dosn't exist, so it's no-N3.

   Looking for pre-exisiting NP with item 1-PHENYLETHANOL.
   Found NP predicate for primary-amine, will it succeed?
         Attempting NP using 1-PHENYLETHANOL.....
Looking for a Primary Amine (NRH2) in 1-PHENYLETHANOL:
   Looking for pre-exisiting NP, but having it fail.
     Found NP predicate for primary-amine, will it fail?
         Attempting NP using 1-PHENYLETHANOL.....
Looking for a Primary Amine (NRH2) in 1-PHENYLETHANOL:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

   Looking for pre-exisiting COOH with item 1-PHENYLETHANOL.
   Found COOH predicate for carboxylic-acid, will it succeed?
         Attempting COOH using 1-PHENYLETHANOL.....
Looking for a Carboxylic acid (XC=OOH) in 1-PHENYLETHANOL:
  Found Oxygen at atom 9.  Is it a C=O?
   Looking for pre-exisiting COOH, but having it fail.
     Found COOH predicate for carboxylic-acid, will it fail?
         Attempting COOH using 1-PHENYLETHANOL.....
Looking for a Carboxylic acid (XC=OOH) in 1-PHENYLETHANOL:
  Found Oxygen at atom 9.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
                (molecule name is ->1-PHENYLETHANOL<- )
p-10-11   WITH A PROBABILITY OF   82.4384
p-13   WITH A PROBABILITY OF   74.7113
p-4   WITH A PROBABILITY OF   54.8416
p-16   WITH A PROBABILITY OF   54.8416
p-8-9   WITH A PROBABILITY OF   34.9719
p-12   WITH A PROBABILITY OF   34.9719
p-15   WITH A PROBABILITY OF   -16.1216
p-5-7   WITH A PROBABILITY OF   -53.0281
p-14   WITH A PROBABILITY OF   -68.64

The following is the data for (choose-a-chiral : B A A A B B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
   YES
?-
?- run
```

```
What is the name of the rule base you want to use?
      Please enter it here -> chiral-b

The Rulebase ->chiral-b<- is already loaded.


In looking at choose-a-chiral, trying to determine a-column,
     what molecule would you like to investigate?
     Enter the molecule name here --> 1-PHENYLETHYLAMINE
  [1-PHENYLETHYLAMINE] currently exists in runtime Molecule database.....


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
        (molecule name is ->1-PHENYLETHYLAMINE<- )

   Looking for pre-exisiting CdO with item 1-PHENYLETHYLAMINE.
   Found CdO predicate for carbonyl, will it succeed?
         Attempting CdO using 1-PHENYLETHYLAMINE.....
Looking for a C=O in 1-PHENYLETHYLAMINE:
   Looking for pre-exisiting CdO, but having it fail.
     Found CdO predicate for carbonyl, will it fail?
         Attempting CdO using 1-PHENYLETHYLAMINE.....
Looking for a C=O in 1-PHENYLETHYLAMINE:
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

   Looking for pre-exisiting has-benz with item 1-PHENYLETHYLAMINE.
   Looking for pre-exisiting has-benz, but having it fail.
   Looking for pre-exisiting opposite of has-benz.
WHICH OF THE FOLLOWING IS TRUE ABOUT    aromatic-ring
      A] has-benz
      B] no-benz
Please type in your response as A/B ->  A
*  You chose A, which was has-benz.

   Looking for pre-exisiting OsH with item 1-PHENYLETHYLAMINE.
   Found OsH predicate for hydroxyl, will it succeed?
         Attempting OsH using 1-PHENYLETHYLAMINE.....
Looking for a -OH in 1-PHENYLETHYLAMINE:
   Looking for pre-exisiting OsH, but having it fail.
     Found OsH predicate for hydroxyl, will it fail?
         Attempting OsH using 1-PHENYLETHYLAMINE.....
Looking for a -OH in 1-PHENYLETHYLAMINE:
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

   Looking for pre-exisiting N3 with item 1-PHENYLETHYLAMINE.
   Found N3 predicate for amine, will it succeed?
         Attempting N3 using 1-PHENYLETHYLAMINE.....
Looking for an Amine in 1-PHENYLETHYLAMINE:
*    Found Amine (-NR2) in 1-PHENYLETHYLAMINE at atom 3.
         Executed N3 using 1-PHENYLETHYLAMINE.
*   For amine, Found that N3 exists. [1-PHENYLETHYLAMINE]

   Looking for pre-exisiting NP with item 1-PHENYLETHYLAMINE.
   Found NP predicate for primary-amine, will it succeed?
         Attempting NP using 1-PHENYLETHYLAMINE.....
Looking for a Primary Amine (NRH2) in 1-PHENYLETHYLAMINE:
*    Found Primary Amine (-NH2) in 1-PHENYLETHYLAMINE at atom 3.
         Executed NP using 1-PHENYLETHYLAMINE.
*   For primary-amine, Found that NP exists. [1-PHENYLETHYLAMINE]

   Looking for pre-exisiting COOH with item 1-PHENYLETHYLAMINE.
   Found COOH predicate for carboxylic-acid, will it succeed?
         Attempting COOH using 1-PHENYLETHYLAMINE.....
Looking for a Carboxylic acid (XC-OOH) in 1-PHENYLETHYLAMINE:
   Looking for pre-exisiting COOH, but having it fail.
     Found COOH predicate for carboxylic-acid, will it fail?
         Attempting COOH using 1-PHENYLETHYLAMINE.....
Looking for a Carboxylic acid (XC=OOH) in 1-PHENYLETHYLAMINE:
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
          (molecule name is ->1-PHENYLETHYLAMINE<- )
p-12   WITH A PROBABILITY OF   90.1655
p-13   WITH A PROBABILITY OF   74.7113
p-5-7   WITH A PROBABILITY OF   74.7113
p-10-11   WITH A PROBABILITY OF   54.8416
p-8-9   WITH A PROBABILITY OF   34.9719
p-14   WITH A PROBABILITY OF   19.36
p-16   WITH A PROBABILITY OF   5.5616
p-4   WITH A PROBABILITY OF   -16.1216
p-15   WITH A PROBABILITY OF   -82.4384

The following is the data for (choose-a-chiral : B A B A A B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose no-COOH.
   YES
?-
?-
?-
```

APPENDIX B: OPTIMER Rulebase Generated by KEY

Appendix B contains the rulebase for the OPTIMER system introduced in Text Frame 24. It consists of three sections: some general information about the rulebase, followed by a list of the traits, and lastly the rules themselves. Consult Section IV.B and IV.E for a thorough discussion of the format of the rulebase and how it is generated. The Prolog inference engine that inputs this rulebase can be found in Appendix E.

```
;                    ******* File CHIRAL.HC *******

(assert
; generated using PCT

((quest lat "chiral" "choose-a-chiral" "a-column" "functional-grou" "molecule" (
     (carbonyl CdO no-CdO)
     (aromatic-ring has-benz no-benz)
     (hydroxyl OsH no-OsH)
     (amine N3 no-N3)
     (primary-amine NP no-NP)
     (carboxylic-acid COOH no-COOH)
          )     ))

((maxrules 108))

((chiral 1 p-4 44) if (fact carbonyl CdO))
((chiral 2 p-4 -44) if (fact carbonyl no-CdO))
((chiral 3 p-5-7 44) if (fact carbonyl CdO))
((chiral 4 p-5-7 -44) if (fact carbonyl no-CdO))
((chiral 5 p-8-9 44) if (fact carbonyl CdO))
((chiral 6 p-8-9 -44) if (fact carbonyl no-CdO))
((chiral 7 p-10-11 -44) if (fact carbonyl CdO))
((chiral 8 p-10-11 44) if (fact carbonyl no-CdO))
((chiral 9 p-12 -44) if (fact carbonyl CdO))
((chiral 10 p-12 44) if (fact carbonyl no-CdO))
((chiral 11 p-13 -44) if (fact carbonyl CdO))
((chiral 12 p-13 44) if (fact carbonyl no-CdO))
((chiral 13 p-4 44) if (fact aromatic-ring has-benz))
((chiral 14 p-4 -44) if (fact aromatic-ring no-benz))
((chiral 15 p-5-7 44) if (fact aromatic-ring has-benz))
((chiral 16 p-5-7 -44) if (fact aromatic-ring no-benz))
((chiral 17 p-8-9 44) if (fact aromatic-ring has-benz))
((chiral 18 p-8-9 -44) if (fact aromatic-ring no-benz))
((chiral 19 p-10-11 44) if (fact aromatic-ring has-benz))
((chiral 20 p-10-11 -44) if (fact aromatic-ring no-benz))
((chiral 21 p-12 44) if (fact aromatic-ring has-benz))
((chiral 22 p-12 -44) if (fact aromatic-ring no-benz))
((chiral 23 p-13 44) if (fact aromatic-ring has-benz))
((chiral 24 p-13 -44) if (fact aromatic-ring no-benz))
((chiral 25 p-14 -44) if (fact aromatic-ring has-benz))
((chiral 26 p-14 44) if (fact aromatic-ring no-benz))
((chiral 27 p-15 -44) if (fact aromatic-ring has-benz))
((chiral 28 p-15 44) if (fact aromatic-ring no-benz))
((chiral 29 p-16 -44) if (fact aromatic-ring has-benz))
((chiral 30 p-16 44) if (fact aromatic-ring no-benz))
((chiral 31 p-4 44) if (fact hydroxyl OsH))
((chiral 32 p-4 -44) if (fact hydroxyl no-OsH))
((chiral 33 p-5-7 -44) if (fact hydroxyl OsH))
((chiral 34 p-5-7 44) if (fact hydroxyl no-OsH))
((chiral 35 p-8-9 -44) if (fact hydroxyl OsH))
((chiral 36 p-8-9 44) if (fact hydroxyl no-OsH))
((chiral 37 p-10-11 44) if (fact hydroxyl OsH))
((chiral 38 p-10-11 -44) if (fact hydroxyl no-OsH))
((chiral 39 p-12 -44) if (fact hydroxyl OsH))
((chiral 40 p-12 44) if (fact hydroxyl no-OsH))
((chiral 41 p-13 -44) if (fact hydroxyl OsH))
((chiral 42 p-13 44) if (fact hydroxyl no-OsH))
((chiral 43 p-5-7 44) if (fact amine N3))
((chiral 44 p-5-7 -44) if (fact amine no-N3))
((chiral 45 p-8-9 -44) if (fact amine N3))
((chiral 46 p-8-9 44) if (fact amine no-N3))
((chiral 47 p-12 44) if (fact amine N3))
((chiral 48 p-12 -44) if (fact amine no-N3))
((chiral 49 p-13 -44) if (fact amine N3))
((chiral 50 p-13 44) if (fact amine no-N3))
((chiral 51 p-14 44) if (fact primary-amine NP))
((chiral 52 p-14 -44) if (fact primary-amine no-NP))
((chiral 53 p-15 -44) if (fact primary-amine NP))
((chiral 54 p-15 44) if (fact primary-amine no-NP))
((chiral 55 p-16 -44) if (fact primary-amine NP))
((chiral 56 p-16 44) if (fact primary-amine no-NP))
((chiral 57 p-15 44) if (fact carboxylic-acid COOH))
((chiral 58 p-15 -44) if (fact carboxylic-acid no-COOH))
((chiral 59 p-16 -44) if (fact carboxylic-acid COOH))
((chiral 60 p-16 44) if (fact carboxylic-acid no-COOH))

)
;  *** END OF assert
```

APPENDIX C: KEY Source Code

Appendix C contains source listings for KEY, the automated Knowledge Extraction utilitY. The source code, written in Microsoft C, is broken down into 5 modules: GRIDEFS6.H, KEY6.C, GRID6.C, ANALY6.C, and NEW6.C. Section VII.A.4 explains each of these modules, and Section VII.C.1 explains the environment used to develop them.

```
/*******************************************************/
/*****                                             *****/
/*****      GRIDEFS6.H - include file for KEY6.C     *****/
/*****                April 8, 1992                *****/
/*****                                             *****/
/*******************************************************/

#include "stdio.h"
#define MAX_CMDS 11   /* max num of commands that there can be in1 list */
#define CMDSTRNG 100  /* max size of string of commands sent make_cmd_list */
#define NAME_LEN 16   /* size of label, name, name2 for elements & traits */
                      /*   NOTE - all printf's still use %15s for printing, */
                      /*   and max_position calculation uses 15 as well */
typedef struct _cell {
  int value;
  struct _cell *next, *last;
  } CELL;

typedef struct _title {
  char label[NAME_LEN]; /* label describes name & name2-used only for traits */
  int weight;           /* holds how important the expert thinks a trait is */
  char name[NAME_LEN];
  char name2[NAME_LEN];/*elems abbr only uses first 3,trats opposite uses all*/
  struct _cell *grid;  /* only used by trait list to point to grid of values */
  struct _title *next, *last;
  } TITLE;

typedef struct _item {
  int number;
  int opsize;   /* traits store size of element list used to create the grid */
  char type[9], type2[15], ext[5];  /* element-trait, abbr-opp, ELM-TRT-GRD */
  char filename[14];  /* holds the name of the file that was input */
  struct _title *head;     /* pointer to head of doubly linked list */
  } ITEM;

typedef struct _match {
  int level;      /* the %(0-100) level a correlation is reported as a match */
  int dble_num, dblt_num;   /* the number of correlation matches occuring */
  struct _double *dbl_e, *dbl_t;  /* pointers to correlation match lists */
  int clstre_num, clstrt_num; /* number of cluster analysis matches occuring */
  struct _double *clstr_e, *clstr_t; /*pointers to cluster analy match lists */
  } MATCH;

typedef struct _double {
  int item_a, item_b;     /* the numbers of the two items that matched */
  int score;              /* the % (0-100) level that the two items matched */
  struct _double *next;   /* pointer to the next pair of items that matched */
  } DOUBLE;

   /*****          -=<* Directory of Procedures : KEY6.C *>=-          *****/

void display_size (void);
void make_cmd_list (char [CMDSTRNG], int *, char [MAX_CMDS][10]);
int check4cmd (char [30], char [MAX_CMDS][10], int);
void help_cmd_list (char [10], int, char [MAX_CMDS][10]);
void make_elem_list (void);
void in_elem_list (void);
void make_trat_list (void);
void in_trat_list (void);
void three_diff_nums (int, int *, int *, int *);
void make_grid (void);
void in_grid (void);


   /*****          -=<* Directory of Procedures - File GRID6.C *>=-       *****/

void list_item (ITEM *);
void edit_item_list (ITEM *);
void change_item (ITEM *);
void get_name (char *, char *);
void ch_item_name (ITEM *, TITLE *, int);
void add_item (ITEM *);
void add_node (ITEM *, TITLE *, int);
void remove_item (ITEM *);
void remove_node (ITEM *, TITLE *, int);
void del_item_list (ITEM *);
void del_elem_grid (int);
void del_trat_grid (CELL *);
char get_filename (char *, char[5]);
void save_item_list (ITEM *, int);
void get_item_list (ITEM *, int);
void list_grid (void);
void grid_elem_abbr (int *);
void grid_heading (int *);
void grid_trat_valu (TITLE *, int *);
void ch_elem_cells (TITLE *, int);
void ch_trat_cells (TITLE *);
void add_elem_cells (int);
void add_trat_cells (TITLE *);

   /*****          -=<* Directory of Procedures - File ANALY6.C *>=-       *****/

void analyze_grid (void);
void save_all (void);
void anal_egrid (void);
void anal_tgrid (void);
void corr_grid_head (ITEM *, int *);
void ecorr_grid (void);
void tcorr_grid (void);
void ecluster_grid (void);
void tcluster_grid (void);
void add_match_list (DOUBLE **, int, int, int);
void del_match_list (DOUBLE **);
void del_all_matches (void);
void report_ematch (DOUBLE *);
void report_tmatch (DOUBLE *);
void ch_match_value (void);
void rate_traits (void);

   /*****          -=<* Directory of Procedures - File NEW6.C *>=-       *****/

void rule_gen (void);
void ch2_t_prolog (char *);
void ch2_hc_prolog (char *);
int scale_factor (int);
void gen_t_prolog (void);
void gen_hc_prolog (void);
void gen_prolog (void);
int auto_anal (void);
char *find_name (ITEM *, int);
char *find_name2 (ITEM *, int);
char *find_label (ITEM *, int);
void combine_elem (DOUBLE *);
void verify_elem (DOUBLE *);
void verify_trat (DOUBLE *);


#include "GRIDEFS6.H"
#include <sys/types.h>     /* these three definations are used to generate */
#include <sys/timeb.h>     /*    random numbers for in_trat_list()         */
#define MAX_RND 32767      /* max random number that can be generated      */

ITEM *elem, *trat;
MATCH *m_list;
int disp_r, disp_c;

/*                -=<* Directory of Procedures : KEY6.C *>=-
*      -=<* For Other Procedures, See Files GRID6.C, ANALY6.C, NEW6.C *>=-
```

```
*
********>void display_size()
* FUNCT : determine the size of display being used to set parameters
*         for outputting information to the screen
*
********>void make_cmd_list (cmds, num_cmds, cmd_ary)
* FUNCT : make the array of commands that will be used to check
*         the users input for commands entered
*
********>int check4cmd (line, cmd_ary, num_cmds)
* FUNCT : check the users input to see if a command has been
*         entered
*
********>void help_cmd_list (name, num_cmds, cmd_ary)
* FUNCT : print out the array of commands understood by
*         the calling routine
*
********>void make_elem_list()
* FUNCT : create the element list
*
********>void in_elem_list()
* FUNCT : input all of the elements for use in the  construct
*
********>void make_trat_list()
* FUNCT : create the trait list
*
********>void in_trat_list()
* FUNCT : input all of the traits and their opposites for
*         use in the construct
*
********>void three_diff_nums (num_elem, num1, num2, num3)
* FUNCT : randomly select three unique element numbers
*
********>void make_grid()
* FUNCT : create the grid of values
*
********>void in_grid()
* FUNCT : input the trait values for each element
*
*/


/*                              -=<* TO DO *>=-
** - change get&save_item_list to use cmd_list's at filename prompt.
**    Commands would be HELP, DONE, DIR (take dir of default drive using
**    default extension unless specified otherwise)  Add ability to
**    change directory where files are looked for.
** - get rid of head node of item list by following the way the trait
**    grid is implemented - see part of del_item_list where grid is deleted.
** - might want to store the 3 elements used to elicit the trait names
** - after you analyze the elements, you might want to add traits to remove
**    similarities in stead of adding more elements
** - add an APPEND option for inputting trait and element lists
** - maybe change all number inputting to ASCII inputting so you could detect
**    a CRET instead of having to check numbers.
** - add ability to save trait weights with the grid.  Use a file extension
**    other than ELM, TRT, GRD, RUL --
*/

/*************************************************************************>main()
* NOTES : ???
*  1)   Spacing of output: 2 blank lines between seperate functions, 1 blank
*       line between output within a function OR when sub-functions are called.
*       Spacing setup should be done at the END of the called function, so all
*       functions can begin output with no blank lines.
*/
main()
{
char line[30], cmd_ary[MAX_CMDS][10];
int count, index, num_cmds, cmd_num, elemnum, tratnum;
TITLE *newnode;

   /* initialize the doubly linked element list - head element won't be used */
elem = (ITEM *)malloc(sizeof(ITEM));
elem->number = 0;
elem->opsize = 0;
strcpy(elem->type,"element");
strcpy(elem->type2,"abbreviation");
strcpy(elem->ext,".ELM");
elem->filename[0] = NULL;
newnode = (TITLE *)malloc(sizeof(TITLE));
elem->head = newnode;
newnode->grid = NULL;
newnode->next = newnode;
newnode->last = newnode;
newnode->label[0] = NULL;
newnode->weight = 0;
strcpy(newnode->name,"element head");
strcpy(newnode->name2,"element head 2");
   /* initialize the doubly linked trait list - head trait won't be used */
trat = (ITEM *)malloc(sizeof(ITEM));
trat->number = 0;
trat->opsize = 0;
strcpy(trat->type,"trait");
strcpy(trat->type2,"opposite");
strcpy(trat->ext,".TRT");
trat->filename[0] = NULL;
newnode = (TITLE *)malloc(sizeof(TITLE));
trat->head = newnode;
newnode->grid = NULL;
newnode->next = newnode;
newnode->last = newnode;
newnode->label[0] = NULL;
newnode->weight = 0;
strcpy(newnode->name,"trait head");
strcpy(newnode->name2,"trait head 2");
                              /* initialize the linked match list */
m_list = (MATCH *)malloc(sizeof(MATCH));
m_list->level = 90;       /* start match % at this level */
m_list->dble_num = 0;
m_list->dblt_num = 0;
m_list->dbl_e = NULL;
m_list->dbl_t = NULL;
m_list->clstre_num = 0;
m_list->clstrt_num = 0;
m_list->clstr_e = NULL;
m_list->clstr_t = NULL;

printf("\n\n\n\n");
display_size();
make_elem_list();
make_trat_list();
do {
  make_grid();
  if (trat->head->next->grid != NULL) {           /* if the grid exists, */
      analyze_grid();                             /*   then analyze it */
      break;
    }     /* end if */
    else {                              /* if not, see if user wants to quit */
      printf("NOTE   There is currently no grid present.  Remember, you ");
      printf("can either\n     ENTER the grid from the terminal, or GET ");
      printf("the grid from a file.\n     The grid MUST first exist to be ");
      printf("analyzed.  Would you like to exit\n     from the program? ");
      printf("(YES or NO*) -> ");
      gets(line);
      strupr(line);
      printf("\n\n\n");
    }   /* end if else */
```

```
  } while (strstr(line,"YES") == NULL);

del_item_list (elem);
free(elem->head);
free(elem);
del_item_list (trat);
free(trat->head);
free(trat);
del_all_matches();
free(m_list);
}                              /* end main */


/*************************************************************>void display_size()
* FUNCT : determine the size of display being used to set parameters
*         for outputting information to the screen
* VARS  : disp_c <- holds the number of columns(or characters) of the display
*         disp_r <- holds the number of rows(or lines) of the display
* NOTES : ??
*/
void display_size()
{
char line[30];
int disp_num;

  AGAIN:
  printf("Below are listed the display modes that are currently available ");
  printf("for use.  The\n     numbers indicate columns(or characters) by ");
  printf("rows(or lines) currently\n     being shown by this display:\n\n");
  printf("                          1)  80 x 25 - Any Display\n\n");
  printf("                          2)  80 x 43 - Enhanced Display\n\n");
  printf("                          3) 132 x 25 - Multi Display\n\n");
  printf("                          4) 132 x 43 - Multi Display\n\n");
  printf("                          5)  80 x 60 - Multi Display\n\n");
  printf("                          6) 120 x 25 - Enhanced Display\n\n");
  printf("                          7) 120 x 43 - Enhanced Display\n\n");
  printf("                          8) 132 x 25 - Color Display\n\n");
  printf("Please enter the number corresponding to the current display size: ");
  scanf("%d", &disp_num);
  gets(line);                  /* clear newline character from input */
  printf("\n\n");
  switch (disp_num) {
      case 1: disp_c = 80;
              disp_r = 25;
              break;
      case 2: disp_c = 80;
              disp_r = 43;
              break;
      case 3: disp_c = 132;
              disp_r = 25;
              break;
      case 4: disp_c = 132;
              disp_r = 43;
              break;
      case 5: disp_c = 80;
              disp_r = 60;
              break;
      case 6: disp_c = 120;
              disp_r = 25;
              break;
      case 7: disp_c = 120;
              disp_r = 43;
              break;
      case 8: disp_c = 132;
              disp_r = 25;
              break;
      default:printf("ERROR - Please re-enter the display size!\n\n");
              goto AGAIN;
  }    /* end switch */

  printf("Your display should display %d columns and %d rows.\n\n\n\n\n\n",
      disp_c, disp_r);
}                          /* end display_size() */


/*********************************>void make_cmd_list (cmds, num_cmds, cmd_ary)
* FUNCT : make the array of commands that will be used to check
*         the users input for commands entered
* PARMS : cmds -> string of allowed commands, all in CAPS
*         num_cmds <- a pointer to the total number of commands
*             need to send the address (use &) of the variable
*         cmd_ary <- the array of commands
* NOTES : ??
*/
void make_cmd_list (cmds, num_cmds, cmd_ary)
char cmds[CMDSTRNG], cmd_ary[MAX_CMDS][10];
int *num_cmds;
{
int space, index, cmdlen, pntr;
char cmd_line[CMDSTRNG];

  strcpy(cmd_line,cmds); /* use a copy of string so to not change original */
  pntr = *num_cmds = 0;
  cmdlen = strlen(cmd_line);
  do {                              /* make the command array */
    space = strcspn (cmd_line, " \0");
    for (index = pntr;index < space;index++)
        cmd_ary[*num_cmds][index-pntr] = cmd_line[index];
    cmd_ary[*num_cmds][index-pntr] = NULL;
    *num_cmds = *num_cmds + 1;          /*if space is eol, don't replace*/
    if (cmd_line[space] == ' ') cmd_line[space] = '.';
    pntr = ++space;
    } while (space < cmdlen);                    /* end do while */
/*  help_cmd_list ("", *num_cmds, cmd_ary);*/  /* add only for testing */
}             /* end make_cmd_list (cmds, num_cmds, cmd_ary) */


/*************************************>int check4cmd (line, cmd_ary, num_cmds)
* FUNCT : check the users input to see if a command has been entered
*         The order of commands in the SWITCH statement needs to be
*         the same as the order set up in the routine "make_cmd_list"
* RETNS : a number representing the position of the command in the command
*         array that the users input matched (cmdnum = 0 means no match
*         occured, hence it is assumed a command was not entered)
* PARMS : line -> string containing the users input
*         cmd_ary -> the array of allowable commands
*         num_cmds -> the number of commands in 'cmd_ary'
* NOTES : ??
*  1) - If the user inputs something shorter that four characters,
*       and those few characters match part of one of the the strings
*       commands that is being looked for, it will cause a match with
*       that command.  EX/ inputting "one" will match with "done"
*  2) - If nothing was typed, and just a RETURN was pressed, cmd_num = 1
*       will be returned.  This is because a string of "nothing" will match
*       with any string, hence the first string, command 1, is returned.
*/
int check4cmd (line, cmd_ary, num_cmds)
char line[30], cmd_ary[MAX_CMDS][10];
int num_cmds;
{
int index;
char c_line[30];

  printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
  if (line[0] == NULL) return (0);  /* return '0' if no command was entered */
  strcpy(c_line,line);
  strupr(c_line);
  for (index = 0;index < num_cmds;index++)
      if (strstr(cmd_ary[index],c_line) != NULL)
```

```
        return (index + 1);     /* return number only if command is found */
  return (0);                   /* return '0' if no match occured */
}               /* end check4cmd (line, cmd_ary, num_cmds) */


/*********************************>void help_cmd_list (name, num_cmds, cmd_ary)
* FUNCT : print out the array of commands understood by
*         the calling routine - The order of commands in
*         cmd_ary is not important
* PARMS : name -> either "element" or "trait", depending on
*             what item is being dealt with when help is requested
*         num_cmds -> the number of commands in cmd_ary
*         cmd_ary -> the command array
* NOTES : ??
*/
void help_cmd_list (name, num_cmds, cmd_ary)
char name[10], cmd_ary[MAX_CMDS][10];
int num_cmds;
{
int index;

  printf("There are %d commands you can enter, and they are:\n", num_cmds);
  for (index = 0;index < num_cmds;index++)  /* list info on appropriate cmds */
    {
      if (strstr(cmd_ary[index],"GET") != NULL)
        { printf("    %s - retrieve a saved %s list\n", cmd_ary[index], name);
          continue;  }
      if (strstr(cmd_ary[index],"ENTER") != NULL)
        { printf("    %s - enter a new %s list\n",
             cmd_ary[index], name);
          continue;  }
      if (strstr(cmd_ary[index],"EDIT") != NULL)
        {
          if (strstr(cmd_ary[index],"EDIT-E") != NULL)
            { printf("    %s - edit the list of elements\n", cmd_ary[index]);
              continue;  }
          if (strstr(cmd_ary[index],"EDIT-T") != NULL)
            { printf("    %s - edit the list of traits\n", cmd_ary[index]);
              continue;  }
          printf("    %s - edit the list of %ss\n", cmd_ary[index], name);
          continue;
        }     /* end of "EDIT" */
      if (strstr(cmd_ary[index],"DEL") != NULL)
        { printf("    %s - delete the list of %ss\n", cmd_ary[index], name);
          continue;  }
      if (strstr(cmd_ary[index],"SAVE") != NULL)
        {
          if (strstr(cmd_ary[index],"SAVE-ALL") != NULL)
            { printf("    %s - save the elements, traits, and grid.\n",
                 cmd_ary[index]);
              continue;  }
          if (strstr(cmd_ary[index],"E-SAVE") != NULL)
            { printf("    %s - save the list of elements\n", cmd_ary[index]);
              continue;  }
          if (strstr(cmd_ary[index],"T-SAVE") != NULL)
            { printf("    %s - save the list of traits\n", cmd_ary[index]);
              continue;  }
          printf("    %s - save the list of %ss\n", cmd_ary[index], name);
          continue;
          }    /* end if "SAVE" */
      if (strstr(cmd_ary[index],"CHANGE") != NULL)
        { printf("    %s - change one %s in the list of %ss\n",
             cmd_ary[index], name, name);
          continue;  }
      if (strstr(cmd_ary[index],"ADD") != NULL)
        { printf("    %s - add one %s to the list of %ss\n",
             cmd_ary[index], name, name);
          continue;  }
      if (strstr(cmd_ary[index],"REMOVE") != NULL)
        { printf("    %s - remove one %s from the list of %ss\n",
             cmd_ary[index], name, name);
          continue;  }
      if (strstr(cmd_ary[index],"LIST") != NULL)
        {
          if (strstr(cmd_ary[index],"LIST-E") != NULL)
            { printf("    %s - list out the elements\n", cmd_ary[index]);
              continue;  }
          if (strstr(cmd_ary[index],"LIST-T") != NULL)
            { printf("    %s - list out the triats\n", cmd_ary[index]);
              continue;  }
          if (strstr(cmd_ary[index],"LIST-G") != NULL)
            { printf("    %s - list out the grid\n", cmd_ary[index], name);
              continue;  }
          printf("    %s - list out the %ss\n", cmd_ary[index], name);
          continue;
          }       /* end if "LIST" */
      if (strstr(cmd_ary[index],"NEXT") != NULL)
        { printf("    %s - go on to the next triad of %ss\n",
             cmd_ary[index], name);
          continue;  }
      if (strstr(cmd_ary[index],"DONE") != NULL)
        { printf("    %s - finished entering %s list\n", cmd_ary[index], name);
          continue;  }
      if (strstr(cmd_ary[index],"HELP") != NULL)
        { printf("    %s - print out this list of commands\n", cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"CH-MATCH") != NULL)
        { printf("    %s - change the value at which matches are detected\n",
          cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"E-ANAL") != NULL)
        { printf("    %s - analyze the elements for similarities\n",
          cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"T-ANAL") != NULL)
        { printf("    %s - analyze the traits for similarities\n",
          cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"RATE-T") != NULL)
        { printf("    %s - determine the importance of each trait\n",
          cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"RULES") != NULL)
        { printf("    %s - generate rules from the current grid\n",
          cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"AUTO") != NULL)
        { printf("    %s - automatically do the analysis and rule generation\n",
          cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"T-PROLOG") != NULL)
        { printf("    %s - generate a turbo-prolog system\n", cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"HC-PROLOG") != NULL)
        { printf("    %s - generate an hc-prolog system\n", cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"PROLOG") != NULL)
        { printf("    %s - generate an prolog system with entailments\n",
          cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"E-CORR") != NULL)
        { printf("    %s - perform correlation analysis on the elements\n",
          cmd_ary[index]);
          continue;  }
      if (strstr(cmd_ary[index],"T-CORR") != NULL)
        { printf("    %s - perform correlation analysis on the traits\n",
          cmd_ary[index]);
          continue;  }
```

```
    if (strstr(cmd_ary[index],"E-CLUST") != NULL)
      { printf("    %s - perform cluster analysis on the elements\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"T-CLUST") != NULL)
      { printf("    %s - perform cluster analysis on the traits\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"RPT-EMAT") != NULL)
      { printf("    %s - report matches found while analyzing the elements\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"RPT-TMAT") != NULL)
      { printf("    %s - report matches found while analyzing the traits\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"RPT-TMAT") != NULL)
      { printf("    %s - report matches found while analyzing the traits\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"RPT-TMAT") != NULL)
      { printf("    %s - report matches found while analyzing the traits\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"RPT-TMAT") != NULL)
      { printf("    %s - report matches found while analyzing the traits\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"RPT-TMAT") != NULL)
      { printf("    %s - report matches found while analyzing the traits\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"QUIT") != NULL)
      { printf("    %s - exit the automated analysis and generation process\n",
        cmd_ary[index]);
        continue;  }
    if (strstr(cmd_ary[index],"FINISHED") != NULL)
      { printf("    %s - finished with the analysis of the %ss\n",
        cmd_ary[index], name);
        continue;  }
    if (strstr(cmd_ary[index],"COMBINE") != NULL)
      { printf("    %s - combine a pair of %s that are closely matched\n",
        cmd_ary[index], name);
        continue;  }
    if (strstr(cmd_ary[index],"INPUT") != NULL)
      { printf("    %s - input a new %s to ", cmd_ary[index], trat->type);
        printf("better distinguish between two %ss\n", elem->type);
        continue;  }
    if (strstr(cmd_ary[index],"VERIFY") != NULL)
      { printf("    %s   verify all the grid ", cmd_ary[index]);
        printf("values for the two matching %ss\n", name);
        continue;  }
    if (strstr(cmd_ary[index],"GO-ON") != NULL)
      { printf("    %s   consider the next pair of matching %ss\n",
        cmd_ary[index], name);
        continue;  }
    printf("    %s -\n",cmd_ary[index]); /* print out command other  */
                                         /*  than those listed above */
  };       /* end for */
  printf("\n\n");
}                /* end help_cmd_list (item, num_cmds, cmd_ary) */


/*********************************************************>void make_elem_list()
* FUNCT : create the element list
* VARS  : elem -> pointer to the element list
* NOTES : ??
*/
void make_elem_list()
{
char line[30], cmd_ary[MAX_CMDS][10];
int count, index, num_cmds, cmd_num, elemnum, tratnum;

  make_cmd_list ("GET ENTER EDIT DEL SAVE LIST DONE HELP", &num_cmds, cmd_ary);
  if (elem->number == 0) printf("Please enter the names of the elements.\n");
  do  {
    printf("    Your choices for obtaining the element names are as ");
    printf("follows:\n\n");
    help_cmd_list ("element", num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {    /* "GET ENTER EDIT DEL SAVE LIST DONE HELP" */
      case 0 :          /* no commands match with users input */
              printf("The command you entered, %s, is not understood.",
                  line);
              printf("  Please enter\n             your choice again.\n\n\n\n\n");
              break;
      case 1 : get_item_list (elem, 0);
              break;
      case 2 : in_elem_list ();
              break;
      case 3 : edit_item_list (elem);
              break;
      case 4 : printf("Are you SURE you want to DELETE the element list?");
              printf(" (YES or *NO) ");
              gets(line);
              strupr(line);
              printf("\n\n");                       /* break on anything but yes */
              if (strstr(line,"YES") == NULL) {
                  printf("    ABORT - Element list NOT deleted!\n\n\n\n\n");
                  break;
                }
              del_item_list (elem);
              break;
      case 5 : save_item_list (elem, 0);
              break;
      case 6 : list_item (elem);
              break;
      case 7 : break; /* NOTE if this num changes, change DO-WHILE condition */
      case 8 : break;  /* HELP - the command list will print automatically */
     default : printf("ERROR - element command not accounted for!!!");
              gets(line);
              printf("\n\n\n\n");
      }        /* end switch */
    } while (cmd_num != 7);      /* end do - while */
  printf("\n\n");
}                              /* end make_elem_list() */


/*********************************************************>void in_elem_list()
* FUNCT : input all of the elements for use in the
*         construct
* VARS  : elem <-> a pointer to the element list
* NOTES : ??
*  1) - you will not be able to enter an element name with any of the
*       following combinations of letters -- LIST, list, DONE, done, HELP, help.
*       This is due to using the function "strstr" to detect when the
*       user wants to list out the elements, when the user is finished
*       entering elements, or if the user wants help.
*/
void in_elem_list()
{
TITLE *newnode, *prior;
int num_cmds, cmd_num, space_pos;
char line[30], cmd_ary[MAX_CMDS][10];
```

```
    if (elem->head->next != elem->head) {
        printf("ERROR - The element list already exists.  Use EDIT to make ");
        printf("any changes!\n\n\n\n");
        return;
      }          /* end if */
    printf("Now enter the names of the elements, one to a line,");
    printf(" or the name of a command.\n\n");
    make_cmd_list ("LIST DONE HELP", &num_cmds, cmd_ary);
    help_cmd_list ("element",num_cmds, cmd_ary);
    do {
        printf("Name of element %d -> ", elem->number + 1);
        gets(line);
        printf("\n");
/* NOTE - inputting "one" will match with "done" - see note 1 of check4cmd */
        cmd_num = check4cmd (line, cmd_ary, num_cmds);
        switch (cmd_num) {      /* "LIST DONE HELP" */
            case 0 : break;     /* no command entered, element entered */
            case 1 : list_item (elem);
                     continue;
            case 2 : printf("\n\n");
                     return;
            case 3 : help_cmd_list ("element", num_cmds, cmd_ary);
                     continue;
           default : printf("\nERROR in command number == %d %s",
                            cmd_num, line);
                     gets(line);
                     printf("\n\n\n\n");
                     continue;
         }          /* end switch */
        /* add another node to the doubly linked list */
        newnode = (TITLE *)malloc(sizeof(TITLE));
        prior = elem->head->last;
        newnode->last = prior;
        newnode->next = elem->head;
        prior->next = newnode;
        elem->head->last = newnode;
        newnode->grid = NULL;
        newnode->label[0] = NULL;
        newnode->weight = 0;
        elem->number++;
        get_name(newnode->name, line);
        printf("     Three letter abbreviation for element : %s -> ",
            newnode->name);
        gets(line);
        printf("\n\n");
        get_name(newnode->name2, line);
        newnode->name2[3] = NULL;
    } while (cmd_num != 2);          /* end do-while */
}                                    /* end in_elem_list() */


/*************************************************************>void make_trat_list()
* FUNCT : create the trait list
* VARS  : trat < > pointer to the trait list
*         elem -> pointer to the element list
* NOTES : ??
*/
void make_trat_list()
{
char line[30], cmd_ary[MAX_CMDS][10];
int count, index, num_cmds, cmd_num, elemnum, tratnum;

  make_cmd_list ("GET ENTER EDIT DEL SAVE LIST DONE HELP", &num_cmds, cmd_ary);
  if (trat->number == 0) printf("Next, enter the names of the traits.\n");
  do  {
    printf("    Your choices for obtaining the trait names are as follows:\n\n");
    help_cmd_list ("trait", num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {   /* "GET ENTER EDIT DEL SAVE LIST DONE HELP" */
      case 0 :           /* no commands match with users input */
               printf("The command you entered, %s, is not understood.",
                   line);
               printf("  Please enter\n             your choice again.\n\n\n\n");
               break;
      case 1 : get_item_list (trat, elem->number);
               break;
      case 2 : in_trat_list();
               break;
      case 3 : edit_item_list (trat);
               break;
      case 4 : printf("Are you SURE you want to DELETE the trait list?");
               printf(" (YES or *NO) ");
               gets(line);
               strupr(line);
               printf("\n\n");                    /* break on anything but yes */
               if (strstr(line,"YES") == NULL) {
                   printf("    ABORT - Trait list NOT deleted!\n\n\n\n");
                   break;
                 }
               del_item_list (trat);
               break;
      case 5 : save_item_list (trat, 0);
               break;
      case 6 : list_item (trat);
               break;
      case 7 : break; /* NOTE if this num changes, change DO-WHILE condition */
      case 8 : break;  /* HELP - the command list will print automatically */
     default : printf("ERROR - trait command not accounted for!!!");
               gets(line);
               printf("\n\n\n\n");
      }         /* end switch */
    } while (cmd_num != 7);      /* end do - while */
  printf("\n\n");
}                                /* end make_trat_list() */


/*************************************************************>void in_trat_list()
* FUNCT : input all of the traits and their opposites for
*         use in the construct
* VARS  : trat <-> a pointer to the trait list
*         elem -> a pointer to the element list
* NOTES : ??
*/
void in_trat_list()
{
TITLE *prior, *newnode, *elem1, *elem2, *elem3;
struct timeb timebuffer;
char line[30], cmd_ary[MAX_CMDS][10];
int index, cmd_num, num_cmds, rnd_seed, num1, num2, num3, space_pos;

  if (trat->head->next != trat->head) {
      printf("ERROR - The trait list already exists.  Use EDIT to make ");
      printf("any changes!\n\n\n\n");
      return;
    }          /* end if */
  ftime(&timebuffer);      /* use milliseconds on clock as the */
  rnd_seed = (timebuffer.millitm / 10) + 1;
  srand(rnd_seed);         /* seed for the random number generator */
  printf("Now enter the names of the traits, one to a line,");
  printf(" or the name of a command.\n\n");
  make_cmd_list ("LIST NEXT DONE HELP", &num_cmds, cmd_ary);
  help_cmd_list (trat->type, num_cmds, cmd_ary);
  three_diff_nums (elem->number, &num1, &num2, &num3);
  do {
      printf("Consider the following three elements : \n");
```

```
    elem1 = elem->head->next;
    for (index = 0;index < num1;index++) elem1 = elem1->next;
    printf("        %15s\n", elem1->name);
    elem2 = elem->head->next;
    for (index = 0;index < num2;index++) elem2 = elem2->next;
    printf("        %15s\n", elem2->name);
    elem3 = elem->head->next;
    for (index = 0;index < num3;index++) elem3 = elem3->next;
    printf("        %15s\n\n", elem3->name);
    printf("Please determine a trait that two of the above elements have ");
    printf("that the third\n");
    printf("     does not, and enter the name of that trait below.\n\n");
    printf("Name of trait %d -> ", trat->number + 1);
    gets(line);
    printf("\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {  /* "LIST NEXT DONE HELP" */
        case 0 : break;     /* no command entered, element entered */
        case 1 : list_item (trat);
                 continue;
        case 2 : three_diff_nums (elem->number, &num1, &num2, &num3);
                 continue;
        case 3 : printf("\n\n");
                 return;
        case 4 : help_cmd_list ("trait", num_cmds, cmd_ary);
                 continue;
       default : printf("\nERROR in command number == %d %s",
                        cmd_num, line);
                 gets(line);
                 printf("\n\n\n\n");
                 continue;
      }        /* end switch */
          /* add another node to the doubly linked list */
    newnode = (TITLE *)malloc(sizeof(TITLE));
    prior = trat->head->last;
    newnode->last = prior;
    newnode->next = trat->head;
    prior->next = newnode;
    trat->head->last = newnode;
    newnode->grid = NULL;
    trat->number++;
    three_diff_nums (elem->number, &num1, &num2, &num3);
    newnode->label[0] = NULL;
    newnode->weight = 0;
    get_name(newnode->name, line);
    printf("     Now please enter the opposite of %s here -> ",
           newnode->name);
    gets(line);
    printf("\n");
    get_name(newnode->name2, line);
    printf("     Think of a name that can be used to describe this trait, ");
    printf("and \n        enter it here -> ");
    gets(line);
    printf("\n\n");
    get_name(newnode->label, line);
  } while (cmd_num != 3);           /* end do-while */
}                       /* end in_trat_list() */


/*************************>void three_diff_nums (num_elem, num1, num2, num3)
* FUNCT : randomly select three unique element numbers
* PARMS : num_elem -> the number of elements in the element list
*         num1, num2, num3 <- the three element numbers
* NOTES : ??
*/
void three_diff_nums (num_elem, num1, num2, num3)
int num_elem, *num1, *num2, *num3;
{
long tmp;

  tmp = rand();
  *num1 = (tmp * num_elem) / MAX_RND;
  SECOND_NUM:
  tmp = rand();
  *num2 = (tmp * num_elem) / MAX_RND;
  if (*num1 == *num2) goto SECOND_NUM;
  THIRD_NUM:
  tmp = rand();
  *num3 = (tmp * num_elem) / MAX_RND;
  if ((*num3 == *num1) || (*num3 == *num2)) goto THIRD_NUM;
/*printf("The three different numbers are %d, %d, %d.\n",
       *num1, *num2, *num3); */        /* ADD ONLY FOR TESTING */
}             /* end three_diff_nums (num_elem, num1, num2, num3) */


/*************************************************************>void make_grid()
* FUNCT : create the grid of values
* VARS  : trat <-> pointer to the trait list
*         elem -> pointer to the element list
* NOTES : ??
*/
void make_grid()
{
char line[30], cmd_ary[MAX_CMDS][10];
int index, cmd_num, num_cmds;

  make_cmd_list ("GET ENTER EDIT DEL SAVE LIST DONE HELP", &num_cmds,
      cmd_ary);
  if (trat->head->next->grid == NULL)
      printf("Now enter each element's value for every trait.\n");
  do  {
    printf("    Your choices for obtaining the element's trait values are as ");
    printf("follows:\n\n");
    help_cmd_list ("value", num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {    /* "GET ENTER EDIT DEL SAVE LIST DONE HELP" */
      case 0 :            /* no commands match with users input */
               printf("The command you entered, %s, is not understood.",
                   line);
               printf("  Please enter\n             your choice again.\n\n\n\n\n");
               break;
      case 1 : strcpy(trat->ext,".GRD");
               get_item_list (trat, elem->number);
               break;
      case 2 : in_grid();
               break;
      case 3 : if (trat->head->next->grid == NULL) {
                   printf("ERROR - The grid needs to be present before you ");
                   printf("can EDIT it.  Please\n     use GET or ENTER ");
                   printf("first.\n\n\n\n\n");
                   break;
                 }     /* end if */
               AGAIN:
               printf("Would you like to edit the grid values by ELEMENT or ");
               printf("by TRAIT? (please\n     enter ELEMENT or TRAIT) ");
               gets(line);
               printf("\n\n\n\n\n");
               strupr(line);
               if (strstr(line,"ELEM") != NULL) edit_item_list(elem);
                 else if (strstr(line,"TRAI") != NULL) edit_item_list(trat);
                   else goto AGAIN;
               break;
      case 4 : printf("Are you SURE you want to DELETE the trait list?");
               printf(" (YES or *NO) ");
               gets(line);
               strupr(line);
```

```
            printf("\n\n");                    /* break on anything but yes */
            if (strstr(line,"YES") == NULL) {
                printf("    ABORT - Trait list NOT deleted!\n\n\n\n");
                break;
              }
            del_item_list (trat);
            strcpy(trat->ext,".TRT");
            make_trat_list();            /* need to re-enter trait list */
            break;
      case 5 : save_item_list (trat, 0);
            break;
      case 6 : list_grid();
            break;
      case 7 : break;  /* NOTE if this num changes, change DO-WHILE condition */
      case 8 : break;  /* HELP - the command list will print automatically */
     default : printf("ERROR - grid command not accounted for!!!");
            gets(line);
            printf("\n\n\n\n");
      }       /* end switch */
    } while (cmd_num != 7);     /* end do - while */
  printf("\n\n");
}               /* end make_grid() */


/*****************************************************************>void in_grid()
* FUNCT : input the trait values for each element
* VARS  : trat <-> a pointer to the trait list
*         elem -> a pointer to the element list
* NOTES : ??
*  1) - There needs to be at least one element and one trait for the
*       do-while loop conditions to work properly
*  2) - the name of the trait's extension is changed from "TRT" to "GRD" to
*       denote that there is now a grid of values attached to the trait list
*/
void in_grid()
{
TITLE *ctrat;
int period;

  if (trat->opsize != 0) {
      printf("ERROR - The grid of values already exists.  Please use EDIT ");
      printf("to make any changes!\n\n\n\n");
      return;
    }      /* end if */
  strcpy(trat->ext,".GRD");           /* grid now associated with trait list */
  period = strcspn(trat->filename,".\0");       /* append a '.GRD' extension */
  trat->filename[period] = NULL;                /*   to the existing filename */
  strcat (trat->filename,".GRD");
  trat->opsize = elem->number;
  ctrat = trat->head->next;  /* current trait is first trait of trait list */
  do {                  /*    do for all traits     */
      add_trat_cells (ctrat);
      ch_trat_cells (ctrat);
      ctrat = ctrat->next;                          /* point to next trait */
  } while (ctrat != trat->head);              /* repeat for all traits */
  printf("\n\n\n");
}               /* end in_grid() */

/*************************************************************************/

#include "GRIDEFS6.H"

extern ITEM *elem, *trat;
extern MATCH *m_list;
extern int disp_r; disp_c;

/*            -=<* Directory of Procedures - File GRID6.C *>=-
*
********>void list_item (item)
* FUNCT : list out the element or trait list
*
********>void edit_item_list (item)
* FUNCT : edit the element or trait list
*
********>void change_item (item)
* FUNCT : determine the number of the item to change
*
********>void get_name (name, line)
* FUNCT : get input from 'line' if there, replace spaces w/'-', copy to 'name'
*
********>void ch_item_name (item, node, item_num)
* FUNCT : change the title and abbr or opp of the element or trait
*
********>void add_item (item)
* FUNCT : determine where to add the new item to the item list
*
********>void add_node (item, node, item_num)
* FUNCT : add a node to the item list
*
********>void remove_item (item)
* FUNCT : to determine which node to remove from the item list
*
********>void remove_node (item, node, item_num)
* FUNCT : to remove a node from the item list
*
********>void del_item_list (item)
* FUNCT : dis-assemble the doubly linked list and free up the
*         storage it uses - leaves head of linked list
*
********>void del_elem_grid (elem_num)
* FUNCT : delete an certain element's values from the trait grid
*
********>void del_trat_grid (tgrid)
* FUNCT : delete a certain trait's values from the trait grid
*
********>char get_filename (filename, file_ext)
* FUNCT : returns a new filename if default sent it was not the correct one
*
********>void save_item_list (item, sav_grid)
* FUNCT : to save an existing element or trait list to a file
*
********>void get_item_list (item, numelem)
* FUNCT : retrieve the element or trait list from a file
*
********>void list_grid()
* FUNCT : list the grid to the screen
*
********>void grid_elem_abbr (lines)
* FUNCT : list the abbreviations and names of the elements to the screen
*
********>void grid_heading (lines)
* FUNCT : list the grid heading to the screen
*
********>void grid_trat_valu (current_item, lines_out)
* FUNCT : list a trait, its opposite and associated values on the screen
*
********>void ch_elem_cells (celem, item_num)
* FUNCT : change the cell values for all elements of a trait
*
********>void ch_trat_cells (ctrat)
* FUNCT : change the cell values for all traits of an element
*
********>void add_elem_cells (item_num)
* FUNCT : add all of the cells for the element number passed
*
********>void add_trat_cells (ctrat)
* FUNCT : add all of the cells for a newly entered trait
*
```

```
*/


/*************************************************************>void list_item (item)
* FUNCT : list out the element or trait list
* PARMS : item -> passed a pointer to the item list
* NOTES : ??
*  1) - This procedure has some special spacing because it is called from
*       command lines that are expecting both commands as well as element or
*       trait names to be input.  It leaves 2 blank lines before it starts
*       printing, and leaves 2 blank lines when it is through printing.
*/
void list_item (item)
ITEM *item;
{
int lines_out, item_num, full_page;
char line[30];
TITLE *node;

  printf("\n\nThere are %d %ss currently.  ", item->number, item->type);
  if (item->number != 0 ) printf("Their names follow :");
  printf("\n\n");
  node = item->head->next;
  item_num = 0;
  lines_out = 5;  /* note: 2 for border of DESQ, 3 for 'hit rtn to continue'*/
  full_page = 10;
  while (node != item->head) {
      if ((lines_out >= disp_r) > 0) { /* so the list doesn't scroll off screen */

          printf("\nPlease hit RETURN to continue, or type QUIT to ");
          printf("stop listing - ");
          gets(line);
          printf("\n\n\n");
          strupr(line);
          if (strstr(line, "QUIT") != NULL) return;
          lines_out = 5;
        }         /* end if */
      printf("The name of %s %d is {%s},", item->type, ++item_num,
          node->name);
      if (node->label[0] == NULL) { /*if no label present-elem or trat w/none*/
          if (disp_c == 80) {/* print CR and spaces if small display is used */
              printf("\n                    ");
              lines_out++;
            }     /* end if */
          printf(" and its %s is {%s}.\n", item->type2,
              node->name2);
          lines_out++;
        }         /* end if */
        else {      /* else only trait list with labels, 2 lines always used */
          printf(" its %s is {%s},\n    and a ", item->type2, node->name2);
          lines_out++;
          printf("description of this %s is {%s}.\n", item->type, node->label);
          lines_out++;
        }         /* end if else */
      node = node->next;   /* move current node pointer to next node of list */
    }           /* end while */
  printf("\nPlease hit RETURN to continue - ");
  gets(line); /* this is so the list doesn't scroll off screen */
  printf("\n\n\n");
}                       /* end list_item (item) */


  /*************************************************>void edit_item_list (item)
  * FUNCT : edit the element or trait list
  * PARMS : item <-> the element or trait array
  * NOTES : ??
  */
  void edit_item_list (item)
  ITEM *item;
{
char line[30], cmd_ary[MAX_CMDS][10];
int num_cmds, cmd_num;

  if (item->number == 0) {        /* make sure the array exists */
      printf("The %s array does not yet exist.  ", item->type);
      printf("Type ENTER to make the array.\n\n\n\n");
      return;
    }     /* end if */
  make_cmd_list ("CHANGE ADD REMOVE LIST LIST-G DONE HELP", &num_cmds, cmd_ary);
  do {
      printf("Your choices for editing the %s list are as follows:\n\n",
          item->type);
      help_cmd_list(item->type,num_cmds,cmd_ary);
      printf("Please enter your choice here -> ");
      gets(line);
      printf("\n\n");
      cmd_num = check4cmd(line, cmd_ary, num_cmds);
      switch (cmd_num) { /* "CHANGE ADD REMOVE LIST LIST-G DONE HELP" */
          case 0 :        /* no commands match with users input */
              printf("The command you entered, %s, is not understood.",
                  line);
              printf("  Please enter\n             your choice again.\n\n");
              break;
          case 1 : change_item (item);/*CHANGE an existing item in list*/
              break;
          case 2 : add_item (item);  /* ADD item either in or at end of list */
              if ((trat->head->next->grid != NULL) &&    /* if grid present, */
                (item == elem))       /* and the item is an element, one must */
                  trat->opsize++;   /* be added to the traits opsize counter */
              break;
          case 3 : remove_item (item);   /* REMOVE a node from the item list */
              if ((trat->head->next->grid != NULL) &&    /* if grid present, */
                (item == elem))    /* and the item is an element, one must be */
                  trat->opsize--;   /* subtracted from traits opsize counter */
              break;
          case 4 : list_item (item);    /* LIST out items */
              break;
          case 5 : list_grid();        /* LIST out the grid */
              break;
          case 6 : /*DONE-NOTE if this num changes, change DO-WHILE condition*/
              break;
          case 7 : break; /* HELP- the command list will print automatically */
          default : printf("ERROR - element command not accounted for!!!");
                    gets(line);
                    printf("\n\n");
        }       /* end switch */
  } while (cmd_num != 6);      /* end do - while */
  printf("\n\n\n");
}                       /* end edit_item_list (item) */


/*************************************************>void change_item (item)
* FUNCT : determine the number of the item to change
* PARMS : item <-> the element or trait array
* NOTES : ??
*/
void change_item (item)
ITEM *item;
{
TITLE *node;
char line[30];
int index, item_num;

  AGAIN:
  printf("Which %s number do you want to change? (enter '0' to LIST %ss,\n",
      item->type, item->type);
  printf("      any negative number to EXIT without making changes) ");
  scanf("%d",&item_num);
```

```
  gets(line);   /* clear newline character from input */
  printf("\n\n");
  if (item_num < 0) {          /* if negative, quit and return */
      printf("\n\n");
      return;
    }          /* end if */
  if (item_num > item->number) {
      printf("INPUT ERROR - Please enter your command again.\n\n");
      goto AGAIN;
    }          /* end if */
  if (item_num == 0) {
      printf("\n\n");
      list_item (item);
      goto AGAIN;
    }          /* end if */
  node = item->head;
  for (index = 0;index < item_num;index++) node = node->next;
  ch_item_name (item, node, item_num);
  if (trat->head->next->grid != NULL)/* only change cells if grid is present */
      if (item->head->next->grid != NULL)/* GRID IS PRESENT:if item is trait */
          ch_trat_cells (node);          /* then add the trait cells */
        else ch_elem_cells (node, item_num); /* else add element cells */
  printf("\n\n\n");
}                       /* end change_item (item) */


/*****************************************************>void get_name (name, line)
* FUNCT : get input from 'line' if there, replace spaces w/'-', copy to 'name'
* PARMS : name <- pointer to where the input should go
*         line -> the line of input
* NOTES : ??
*/
void get_name (name, line)
char *name, *line;
{
int space_pos;

if (line[0] != NULL) {/* replace only if something was entered */
    strncpy(name,line,NAME_LEN);
    space_pos = strcspn(name," ");
    while (space_pos <= (NAME_LEN-1)) {    /* remove all spaces from input */
        if (name[space_pos] == NULL) break; /* break if end of input */
        name[space_pos] = '-';
        space_pos = strcspn(name," ");
      }     /* end while */
    name[NAME_LEN-1] = NULL;
  }             /* end if */
}         /* end get_name (name, line) */


/*************************************>void ch_item_name (item, node, item_num)
* FUNCT : change the title and abbr or opp of the element or trait
* PARMS : item -> pointer to the item list
*         node <-> pointer to the item to be changed
*         item_num -> position of item in item list
* NOTES : ??
*  1) - SPACING : no extra lines output!
*/
void ch_item_name (item, node, item_num)
ITEM *item;
TITLE *node;
int item_num;
{
char line[30];
int space_pos;

  printf("The name of %s %d is currently (%s).  What would you\n",
      item->type, item_num, node->name);
  printf("     like it to be? (RETURN to remain unchanged) ");
  gets(line);
  printf("\n\n");
  get_name (node->name, line);
  printf("The %s's %s is currently (%s).  What would you\n",
      item->type, item->type2, node->name2);
  printf("     like it to be? (RETURN to remain unchanged) ");
  gets(line);
  printf("\n\n");
  get_name (node->name2, line);
  if (item == elem)                          /* if item is an element */
      node->name2[3] = NULL;                     /*  use only 3 chars */
  if (item == trat) {                       /* if a trait, */
      printf("The description of %s %d is currently (%s).  What would ",
      item->type, item_num, node->label);
      printf("you\n     like it to be? (RETURN to remain unchanged) ");
      gets(line);
      printf("\n\n");
      get_name (node->label, line);
    }          /* end if */
  printf("\n     The name of %s %d is (%s),\n", item->type, item_num,
      node->name);
  printf("          and its %s is (%s).\n", item->type2, node->name2);
  if (strstr(item->type2,"opposite") != NULL)       /* if item is a trait, */
      printf("     The description of %s %d is (%s),\n", item->type,
          item_num, node->label);          /* print out trait description */
  printf("\n");
}                   /* end ch_item_name (item, node, item_num) */


/*****************************************************>void add_item (item)
* FUNCT : determine where to add the new item to the item list
* PARMS : item <-> pointer to the item list
* NOTES : ??
*/
void add_item (item)
ITEM *item;
{
TITLE *node;
char line[30];
int index, item_num;

  AGAIN:
  printf("You want to add another %s BEFORE what %s number (enter '0' to \n",
      item->type, item->type);
  printf("     LIST %ss, any negative number to EXIT without additions,\n",
      item->type);
  printf("     or any large number to append the %s to the end) -> ",
      item->type);
  scanf("%d",&item_num);
  gets(line);                   /* clear newline character from input */
  printf("\n\n");
  if (item_num < 0) {          /* if negative, quit and return */
      printf("\n\n");
      return;
    }          /* end if */
  if (item_num == 0) {         /* if input zero, LIST item */
      printf("\n\n");
      list_item (item);
      goto AGAIN;
    }          /* end if */
  if (item_num > item->number) item_num = item->number + 1;
  node = item->head;           /* move to node that new node goes BEFORE */
  for (index = 0;index < item_num;index++) node = node->next;
  add_node (item, node, item_num);
  printf("\n\n\n");
}                   /* end add_item (item) */

/*****************************************>void add_node (item, node, item_num)
```

```
* FUNCT : add a node to the item list
* PARMS : item <-> pointer to the item list
*          node -> pointer to the node BEFORE which the new node is to be added
*          item_num -> the position of the item in the item list
* NOTES : ??
*/
void add_node (item, node, item_num)
ITEM *item;
TITLE *node;
int item_num;
{
TITLE *new_node;

  new_node = (TITLE *)malloc(sizeof(TITLE));
  new_node->last = node->last;  /* attach 'new_node' right before 'node' */
  new_node->next = node;
  node->last->next = new_node;
  node->last = new_node;
  ++item->number;          /* item_num now holds the position of the new node */
  new_node->grid = NULL;
  new_node->weight = 0;
  new_node->name[0] = NULL;       /* initialize character fields */
  new_node->name2[0] = NULL;
  new_node->label[0] = NULL;
  ch_item_name (item, new_node, item_num); /*enter item name, and abbr or opp*/
  if (trat->head->next->grid != NULL){/* only change cells if grid is present */
      if (item->head->next->grid != NULL){/*GRID IS PRESENT:if item is trait */
          add_trat_cells (new_node);              /* then add the trait cells */
          ch_trat_cells (new_node);       }       /*  and input values */
       else {
          add_elem_cells (item_num);               /* else add element cells */
          ch_elem_cells (new_node, item_num);      /*  and input values */
       }       /* end if else and end if */
}                           /* end add_node (item) */


/*******************************************************>void remove_item (item)
* FUNCT : to determine which node to remove from the item list
* PARMS : item <-> pointer to the item list
* NOTES : ??
*  1) : if passed a trait list, will delete grid.  if passed element list,
*          will delete that element from the trait grid.
*/
void remove_item (item)
ITEM *item;
{
TITLE *node;
char line[30];
int index, item_num;

  AGAIN:
  printf("Which %s number do you want to remove? (enter '0' to LIST %ss,\n",
      item->type, item->type);
  printf("     any negative number to EXIT without making changes) ");
  scanf("%d",&item_num);
  gets(line);   /* clear newline character from input */
  printf("\n\n");
  if (item_num < 0) {          /* if negative, quit and return */
      printf("\n\n");
      return;
    }         /* end if */
  if (item_num == 0) {             /* if zero, LIST out items */
      printf("\n\n");
      list_item (item);
      goto AGAIN;
    }         /* end if */
  if (item_num > item->number) {  /* if too large, error and try again */
      printf("INPUT ERROR - Please enter your command again.\n\n");
      goto AGAIN;
    }         /* end if */
  node = item->head;                  /* move to node to be removed */
  for (index = 0;index < item_num;index++) node = node->next;
  printf("Are you SURE you want to delete %s %d, whose name is ",
      item->type, item_num);
  printf("'%s' and\n     %s is {%s}", node->name, item->type2, node->name2);
  if (item == trat)                       /* if item is a trait, print out */
      printf(", and is described as\n     {%s}", node->label); /* description */
  printf("? (YES or *NO) ");
  gets(line);
  strupr(line);
  printf("\n\n");
  if (strstr(line,"YES") == NULL) {
      printf("ABORT - %s not deleted!\n\n\n", item->type);
      goto AGAIN;
    }      /* end if */
  remove_node (item, node, item_num); /* remove the item from the item list */
  printf("Single %s in list removed.\n\n\n", item->type);
}                       /* end remove_item (item) */


/*************************************>void remove_node (item, node, item_num)
* FUNCT : to remove a node from the item list
* PARMS : item <-> pointer to the item list
*         node -> pointer to the node to be removed
*         item_num -> the position of the item in the item list
* NOTES : ??
*  1) : if passed a trait list, will delete grid.  if passed element list,
*         will delete that element from the trait grid.
*/
void remove_node (item, node, item_num)
ITEM *item;
TITLE *node;
int item_num;
{
  if (node->grid != NULL) /* for trats, if grid is present, delete it first */
      del_trat_grid (node->grid);
    else                  /* trat list w/no grid, or elem list w or w/out grid */
      if ((item == elem) &&                          /* for an elem list that */
          (trat->head->next->grid != NULL))       /* has a trat list w/grid */
             del_elem_grid (item_num);
  node->last->next = node->next;  /* re-assign pointers to */
  node->next->last = node->last;  /*  remove node from list */
  free(node);
  item->number--;
}                       /* end remove_node (item, node, item_num) */


/*************************************************>void del_item_list (item)
* FUNCT : dis-assemble the doubly linked list and free up the
*         storage it uses - leaves head of linked list
* PARMS : item <-> a pointer to the item list
* NOTES : ??
*  1)  - This routine always removes the last node of the doubly linked list,
*      until all of the nodes but the head are gone.  It will also delete the
*      GRID of values if it is attached to the trait list.
*  2) - This routine can be passed an item list with only the head node
*      on the list, and it will work ok.  However, this situation should
*      never occure.
*  3) - This routine does not include any protection.  The calling routine
*      must check if the user really wants to delete the item list BEFORE-hand.
*/
void del_item_list (item)
ITEM *item;
{
TITLE *newnode, *prior;
CELL *newcell, *priorc;

  if (item->number == 0) {        /* make sure the array exists */
```

```
    printf("The %s array does not yet exist.  ", item->type);
    printf("Type ENTER to make the array.\n\n\n\n");
    return;
  }     /* end if */
 newnode = item->head->last;     /* initialize pointers */
 prior = newnode->last;
 while (prior != newnode)    {
    if (newnode->grid != NULL)    /* if grid present, delete it first */
       del_trat_grid (newnode->grid);
    prior->next = item->head;   /* detach last node */
    item->head->last = prior;
    free(newnode);                  /* delete it */
    newnode = prior; /* re-assign pointers for next iteration */
    prior = newnode->last;
  }             /* end while */
 item->number = 0;                  /* reset item count */
 item->opsize = 0;          /* for traits, reset associated element list size */
                            /* for elements, this is always zero */
 printf("The %s list has been deleted.\n\n\n\n", item->type);
}                           /* end del_item_list (item) */


/*************************************************>void del_elem_grid (elem_num)
* FUNCT : delete an certain element's values from the trait grid
* PARMS : elem_num -> the element number whose values are to be deleted
* VARS  : trat -> a pointer to the trait list
*         elem -> a pointer to the element list
* NOTES : ??
*  1) - can be done either before or after the element is deleted from the
*         element list.
*  2) - will not work if there is only one cell value in the grid
*/
void del_elem_grid (elem_num)
int elem_num;
{
TITLE *ctrat;
CELL *ccell;
int index;

  ctrat = trat->head->next;                 /* start at the first trait */
  while (ctrat != trat->head) {             /* do for all traits */
     ccell = ctrat->grid;
     for (index = 1;index < elem_num;index++)
        ccell = ccell->next;                /* move to cell to be removed */
     ccell->last->next = ccell->next;     /* remove cell */
     ccell->next->last = ccell->last;
     if (ccell == ctrat->grid)               /* if first cell is to be removed */
        ctrat->grid = ccell->next;          /* re-assign grid pointer */
     free(ccell);
     ctrat = ctrat->next;                   /* move to next trait */
   }          /* end while */
}                   /* end del_elem_grid (elem_num) */


/*************************************************>void del_trat_grid (tgrid)
* FUNCT : delete a certain trait's values from the trait grid
* PARMS : tgrid -> a traits pointer to its grid of values
* NOTES : ??
*/
void del_trat_grid (tgrid)
CELL *tgrid;
{
CELL *cell, *priorc;

  cell = tgrid->last;               /* initialize grid pointers */
  priorc = cell->last;
  while (priorc != cell) {
     priorc->next = tgrid;          /* detach last node of grid*/
     tgrid->last = priorc;
     free(cell);                    /* delete it */
     cell = priorc;                 /* re-assign pointers for next pass */
     priorc = cell->last;
  }     /* end while */
 free(tgrid);                       /* delete last node of grid */
 tgrid = NULL;
}                   /* end del_trat_grid (tgrid) */


/*************************************>char get_filename (filename, file_ext)
* FUNCT : returns a new filename if default sent it was not the correct one
* RETNS : 'T' if the name was obtained, 'F' if exited
* PARMS : filename <-> a pointer to the filename
*         file_ext -> the default file extension
* NOTES : ??
*/
char get_filename (filename, file_ext)
char *filename;
char file_ext[5];
{
char line[20], newname[14];
int pntr, period, index;

  strcpy (newname, filename);
  AGAIN:
  if (newname[0] != NULL) {
     printf("    Is %s the correct file name? (*YES or NO) -> ", newname);
     gets(line);
     strupr(line);
     if (strstr(line,"NO") == NULL) {        /* return if it is the correct */
        strcpy(filename, newname);    /* use newname (if NO is NOT typed) */
        printf("\n\n");
        return ('T');
      }     /* end if */
  }     /* end if */
 printf("\n\nIf no extension is entered, %s will be assumed. (type ",
     file_ext);
 printf("RETURN to exit)\n\n");
 printf("    Filename -> ");
 gets(line);
 printf("\n\n");
 if (line[0] == NULL) {      /* return if nothing is typed */
     printf("\n\n\n");
     return ('F');
  }     /* end if */
 pntr = 0;
 period = strcspn(line,".\0");
 for (index=pntr;index<period;index++)       /* build the file name */
     newname[index]=line[index];
 if (line[period] == '.')  {               /* check if exten. was input */
        for (index=0;index<4;index++)     /*  if so, add it to newname */
           newname[period+index]=line[period+index];
        newname[period+index]=NULL;     }
   else
     {   for (index=0;index<4;index++)   /*  if not, add default exten. */
        newname[period+index]=file_ext[index];
      newname[period+index]=NULL;     }
 strupr(newname);
 goto AGAIN;  /* go back until a correct newname is found */
}                   /* end get_filename (filename) */


/*************************************>void save_item_list (item, sav_grid)
* FUNCT : to save an existing element or trait list to a file
* PARMS : item <-> the element or trait array
*         sav_grid  > =0 to save grid if it is present, if not =0 (or =1) if
*                     only trait list is to be saved
* NOTES : ??
*  1) - element and trait lists are saved to a file, 'name' and 'name2' are
```

```
*      saved on the same line.  For trait and grid lists, the traits description
*      follows next on the same line.  If there is a grid present, the grid
*      values associated with that trait are placed on the following line, 30
*      to a line, using extra lines if necessary.
*  2) - When the grid is present and you want to save only the list of traits,
*      the default filename will not be correct.
*/
void save_item_list (item, sav_grid)
ITEM *item;
int sav_grid;
{
TITLE *node;
CELL *gnode;
FILE *fp;
char line[20], filename[14], ans;
int pntr, period, index;

  if (item->number == 0) {          /* make sure the array exists */
      printf("The %s array does not yet exist.  ", item->type);
      printf("Type ENTER to make the array.\n\n\n\n");
      return;
    }     /* end if */
  AGAIN:
  printf("Please enter name of the file to contain the saved %s list.\n",
      item->type);
  if (sav_grid == 0)  {                 /* if grid is to be saved if present */
      ans = get_filename (item->filename, item->ext);    /* get the filename */
      strcpy (filename, item->filename); /* want to change filename if wrong */
    }
    else {                      /* else just want trat list, so use a .TRT ext */
      strcpy (filename, item->filename);     /* don't want to change GRD ext */
      period = strcspn(filename,".\0");             /* find current extension */
      if (filename[period] == '.') {            /* if there is an extension, */
          filename[period] = NULL;                  /* erase it and */
          strcat (filename, ".TRT");                /* append a TRT extension */
        }     /* end if */
      ans = get_filename (filename, ".TRT");     /* get the filename */
    }     /*  end if else */
  if (ans == 'F') return;          /* if the filename was not input, return */
  fp = fopen(filename,"r");     /* if file does not exist, fp = NULL */
  if (fp != NULL) {   /* if a file already exists, give the user a warning */
    fclose(fp);                     /* first close file */
    printf("The file %s already exists.\n    -=NOTE=- continuing will ",
        item->filename);
    printf("destroy anything currently in that file.\n             Do you ");
    printf("still want to use this file? (YES or *NO) -> ");
    gets(line);
    printf("\n\n");
    strupr(line);
    if (strstr(line,"YES") == NULL)   /* if they do not want to use the file */
      goto AGAIN;                     /*   go back and get another filename */
  }               /* end IF */

  fp = fopen(filename,"w");
  fprintf(fp,"%d ", item->number);
  if (sav_grid == 0)                    /* if grid is to be saved if present */
      fprintf(fp,"%d\n%s %s %s\n",      /* use given opsize and extension */
          item->opsize, item->type, item->type2, item->ext);
    else          /* else just want trat list, so use opsize=0 and .TRT ext */
      fprintf(fp,"0\n%s %s .TRT\n", item->type, item->type2);
                  /* else, saving only trait list, use TRT */
  node = item->head->next;
  while (node != item->head) {
      fprintf(fp,"%-16s%-16s%-16s\n", node->name,
          node->name2, node->label);
        /* if GRID is present and grid values are to be saved, save them */
      if ((node->grid != NULL) && (sav_grid == 0)) {
          gnode = node->grid;
          index = 0;  /* index keeps track of 30 grid values per line */
          do {
              fprintf(fp," %d", gnode->value);
              index++;
              if (index > 30) {       /* if more than 30 grid values, */
                  fprintf(fp,"\n");   /*  put on two different lines  */
                  index = 0;
                } /* end if */
              gnode = gnode->next;
            } while (gnode != node->grid);
          fprintf(fp,"\n");
        }           /* end if - GRID is present */
      node = node->next;   /* move current node pointer to next node of list */
    }           /* end while */
  fclose(fp);
  printf("The %s list has been saved.\n\n\n\n\n", item->type);
}                       /* end save_item_list (item, sav_grid) */


/*****************************************>void get_item_list (item, numelem)
* FUNCT : retrieve the element or trait list from a file
* PARMS : item <-> the element or trait array
*         numelem -> when item is a trait array, the number of elements
*                    in the associated element array
* NOTES : ??
*  1) - numelem is used only when two conditions hold - A) when item is the
*      trait array and B) when there is a grid associated with the trait array.
*      numelem is needed to check to make sure that the grid being input is
*      compatible with the element array that is already present.
*/
void get_item_list (item, numelem)
ITEM *item;
int numelem;
{
TITLE *newnode, *prior;
CELL *newcell, *ccell;
FILE *fp;
char *flagc, line[50], ans;
int flag, period, pntr, index;

  if (item->number != 0) {          /* make sure the array exists */
      printf("The %s array already exists.  ", item->type);
      printf("-=NOTE=- retrieving the %s\n", item->type);
      printf("      array will result in destroying the current %s array.\n\n",
          item->type);
      printf("Do you want to proceed? (YES or *NO) ");
      gets(line);
      strupr(line);
      printf("\n\n");
      if (strstr(line,"YES") == NULL) {
          printf("     ABORT - %s list not destroyed!\n\n\n\n\n", item->type);
          return;
        }
      del_item_list (item);         /* destroy current item array */
      item->filename[0] = NULL;
    }     /* end if */
  AGAIN:
  printf("Please enter the name of the file that contains the %s list.\n",
      item->type);
  ans = get_filename (item->filename, item->ext);
  if (ans == 'F') return;                   /* return if so indicated */
  fp = fopen(item->filename,"r");
  if (fp == NULL) {                 /*  If a file does not exist */
      printf("The file %s does not exist.  Please re-enter the file name.\n\n",
          item->filename);
      goto AGAIN;     /* do while there is no valid filename */
    };          /* end if */

  flag = fscanf(fp,"%d %d", &item->number, &item->opsize);
  if ((item->opsize != 0) && (item->opsize != numelem)) {
```

```
    printf("The trait list you are attempting to retrieve has a grid ");
    printf("associated with it\n     that is not compatible with the ");
    printf("element list you are currently using.\n\n\n");
    fclose(fp);
    goto AGAIN;
  };    /* end if */
flagc = fgets(line,20,fp);  /* get rid of newline character in line 1 */
flag = fscanf(fp,"%s %s %s", item->type, item->type2, item->ext);
flagc = fgets(line,20,fp);  /* get rid of newline character in line 2 */
flagc = fgets(line,50,fp); /* get rid of newline character in line 2 */
while (flagc != NULL) {
      /* Add a node to the doubly linked list */
    newnode = (TITLE *)malloc(sizeof(TITLE));
    prior = item->head->last;
    newnode->last = prior;
    newnode->next = item->head;
    newnode->grid = NULL;
    prior->next = newnode;
    item->head->last = newnode;
    newnode->weight = 0;
    for (index = 0;index <= 15;index++)          /* copy name from input */
        newnode->name[index] = line[index];
    for (index = 14;index > 0;index--)           /* find end of string */
        if (newnode->name[index] != ' ') break;
    newnode->name[index+1] = NULL;               /* insert NULL character */
    for (index = 0;index <= 15;index++)          /* copy name2 from input */
        newnode->name2[index] = line[index+16];
    if (item == elem) {          /* if item is an element */
        newnode->name2[3] = NULL;      /* abbreviation is only 3 chars long */
        newnode->label[0] = NULL;      /* and label is not used */
      }          /* end if */
      else {             /* else, it must be a trait */
          for (index = 14;index > 0;index--)         /* find end of string */
              if (newnode->name2[index] != ' ') break;
          newnode->name2[index+1] = NULL;          /* insert NULL character */
          for (index = 0;index <= 15;index++)      /* copy label from input */
              newnode->label[index] = line[index+32];
          for (index = 14;index > 0;index--)         /* find end of string */
              if ((newnode->label[index] != ' ') &&
                  (newnode->label[index] != '\n')) break;
          newnode->label[index+1] = NULL;          /* insert NULL character */
      }          /* end if else */
    if (item->opsize > 0) {          /* if grid is present, read it in */
        ccell = newnode->grid;  /* add new cells to front of list */
        pntr = 0;        /* only 30 grid values per line */
        for (index = 0;index < item->opsize;index++) {
            newcell = (CELL *)malloc(sizeof(CELL));
            if (ccell == NULL) {         /* if no grid, add first cell - The*/
                ccell = newcell;         /*  first time through, both new- */
                ccell->next = ccell;     /*  cell and ccell are equal. This */
                ccell->last = ccell;     /*  block sets up the pointers */
                newnode->grid = ccell;  }
            pntr++;
            flag = fscanf(fp,"%d", &newcell->value);
            newcell->next = ccell->next;
            newcell->last = ccell;
            ccell->next = newcell;
            newnode->grid->last = newcell;
            ccell = newcell;
            if (pntr >= 30) {
                flagc = fgets(line,20,fp); /* get rid of newline char */
                pntr = 0;
              }     /* end if */
          }     /* end for */
        flagc = fgets(line,20,fp); /* get rid of last newline char */
      }     /* end if */
    flagc = fgets(line,50,fp); /* get next line - set flag for test */
          /* end while */
  }
fclose(fp);
```

```
  printf("The %s list has been restored.\n\n\n\n", item->type);
}                /* end get_item_list (item, numelem) */


/*************************************************************>void list_grid()
* FUNCT : list the grid to the screen
* VARS  : trat -> a pointer to the trait list
*         elem -> a pointer to the element list
*         disp_r -> the number of rows the current display uses
* NOTES : ??
*/
void list_grid()
{
TITLE *current_item;
int lines_out;
char line[30];

  if (trat->head->next->grid == NULL) {
      printf("NOTE - The grid cannot be listed out since it does not yet ");
      printf("exist.\n\n\n\n\n");
      return;
    }     /* end if */
  lines_out = 3;  /* note: 3 for 'hit rtn to continue'*/
  grid_elem_abbr (&lines_out);    /* print out the abbrs and names of elements */

  grid_heading (&lines_out);      /* print out the grid heading */
  current_item = trat->head->next;
/* print CR and spaces if small display is used */
  while (current_item != trat->head) { /* print out values and traits */
      if (lines_out >= disp_r) {
          printf("\nPlease hit RETURN to continue, or type QUIT to ");
          printf("stop listing - ");
          gets(line);
          strupr(line);
          if (strstr(line, "QUIT") != NULL) {
              printf("\n\n\n\n");
              return;  }
          printf("\n");
          lines_out = 3;
          grid_heading (&lines_out);
        } /* end if */
      grid_trat_valu (current_item, &lines_out);/*list values and assoc trait*/
      current_item = current_item->next;
    } /* end while */
  printf("\nPlease hit RETURN to continue -");
  gets(line);
  printf("\n\n\n\n");
}                    /* end list_grid (lines_out) */


/*************************************************>void grid_elem_abbr (lines)
* FUNCT : list the abbreviations and names of the elements to the screen
* PARMS : lines <-> a pointer to the number of lines put out to the screen
* VARS  : elem -> a pointer to the element list
*         disp_c -> the number of columns the current display uses
* NOTES : ??
*  1) - Spacing in this function consists of printing on the current
*       line, and printing no blank lines when returning
*/
void grid_elem_abbr (lines)
int *lines;
{
TITLE *current_item;
int index, position, line_num, max_position;

  max_position = (disp_c - 44) / 4;  /*display 2 for 80 col, 10:120, 21:172 */
      /* 15[ name 16'~1,' 1[hopp 14'~5' 12DE3Q border + 2extra sp = 44 */
  line_num = 0;
  if (trat->head->next->weight != 0)  /* if the trait weights are present */
```

```
    max_position--;             /*   leave 1 extra space to print them out */
  current_item = elem->head->next;
  position = 0;
  while (current_item != elem->head) { /* print out abbre's and names */
      if (position >= max_position) {/* fill entire screen before start over */
          printf("\n");
          line_num++;
          (*lines)++;
          position = 0;
        } /* end if */
            /* offset each new line with 2 spaces to make grid more visible */
      for (index = 0;index < line_num;index++) printf("  ");
      for (index = 0;index < position;index++) printf("    "); /* space over */
      printf(" %3s = %s\n", current_item->name2, current_item->name);
      (*lines)++;
      current_item = current_item->next;
      position++;
    } /* end while */
}                       /* end grid_elem_abbr (lines) */


/*****************************************************>void grid_heading (lines)
* FUNCT : list the grid heading to the screen
* PARMS : lines <-> a pointer to the number of lines put out to the screen
* VARS  : trat -> a pointer to the trait list
*         elem -> a pointer to the element list
*         disp_c -> the number of columns the current display uses
* NOTES : ??
*  1) - Spacing in this function consists of printing on the current
*       line, and printing no blank lines when returning
*/
void grid_heading (lines)
int *lines;
{
TITLE *current_item;
int index, position, max_position;

  max_position = (disp_c - 44) / 4;  /* display 9 for 80 col, 19:120, 21:132 */
      /* 15t-name +6'=1,' +15opp +4'=5' +2DESQ border + 2extra sp = 44 */
  if (trat->head->next->weight != 0)  /* If the trait weights are present */
      max_position--;             /*   leave 1 extra space to print them out */
  position = 0;
  current_item = elem->head->next;
  while (current_item != elem->head) { /* print out grid headings */
      if(position >= max_position) { /* fill entire screen before start over */
          printf("\n  ");
          (*lines)++;
          position = 0;
        } /* end if */
      printf(" %3s", current_item->name2);
      current_item = current_item->next;
      position++;
    } /* end while */
  for (index = 0;index < (max_position - position);index++) printf("    ");
  if (trat->head->next->weight != 0)
      printf("(rating) ");
  printf("%10s pairs\n", trat->type);
  (*lines)++;
}                       /* end grid_heading (lines) */


/*********************************>void grid_trat_valu (current_item, lines)
* FUNCT : list a trait, its opposite and associated values on the screen
* PARMS : current_item -> a pointer to the current trait in the trait list
*         lines <-> a pointer to the number of lines printed
* VARS  : disp_c -> the number of columns the current display uses
* NOTES : ??
*  1) - Spacing in this function consists of printing on the current
*       line, and printing no blank lines when returning
*/
void grid_trat_valu (current_item, lines)
TITLE *current_item;
int *lines;
{
CELL *current_cell;
int index, position, max_position;

  max_position = (disp_c - 44) / 4;  /* display 9 for 80 col, 19:120, 21:132 */
      /* 15t-name +6'=1,' +15opp +4'=5' +2DESQ border + 2extra sp = 44 */
  if (current_item->weight != 0)  /* If the trait weights are present */
      max_position--;             /*   leave 1 extra space to print them out */
  current_cell = current_item->grid;
  position = 0;
  do {              /* print grid values*/
      if(position >= max_position) { /* fill entire screen before start over */
          printf("\n  ");
          (*lines)++;
          position = 0;
        } /* end if */
      printf("  %id ", current_cell->value);
      current_cell = current_cell->next;
      position++;
    } while (current_cell != current_item->grid); /* end do-while */
          /* space to side to print trait names */
  for (index = 0;index < (max_position - position);index++) printf("    ");
  if (current_item->weight != 0)
      printf("(%d) ",current_item->weight);
  printf("%s = 5, %s = 1\n", current_item->name, current_item->name2);
  (*lines)++;
}               /* end grid_trat_valu (current_item, lines) */


/*************************************>void ch_elem_cells (celem, item_num)
* FUNCT : change the cell values for all elements of a trait
* PARMS : celem <-> pointer to the element that is to be changed
*         item_num -> position of the element in the element list
* VARS  : trat -> pointer to the list of traits
* NOTES : ??
*  1) - SPACING : no extra lines output!
*/
void ch_elem_cells (celem, item_num)
TITLE *celem;
int item_num;
{
TITLE *ctrat;
CELL *ccell;
char line[30];
int index, number;

  ctrat = trat->head->next; /* current trat is the first trat in list */
  ccell = ctrat->grid;
  printf("\n\nPlease rate the %s {%s} on each of the following traits ",
      elem->type, celem->name);
  printf("using\n      a scale of 1 to 5. (type a negative number to exit, ");
  printf("0 to remain unchanged)\n\n");
  do {              /* do for all traits */
      for (index = 1;index < item_num; index++) /* move to position of */
          ccell = ccell->next;                  /*  the new cell goes */
      printf("Trait {%s} is 5 and {%s} is 1.\n        Its current value is %d",
          ctrat->name, ctrat->name2, ccell->value);
      printf(", the new value is -> ");
      scanf("%d",&number);
      if (number < 0) {       /* type negative number to exit */
          gets(line);
          printf("\n\n\n");
          return;
        }       /* end if */
      if ((number == 0) && (ccell->value == 0)) {
```

```
            printf("     A number between 1 and 5 must be entered.\n");
            continue;
          }     /* end if */
        if ((number < 0) || (number > 5)) {
            printf("     Please enter a number between 1 and 5.\n");
            continue;
          }     /* end if */
        if (number != 0) ccell->value = number; /* put value into new cell */
        ctrat = ctrat->next;            /* set pointer to next trait */
        ccell = ctrat->grid;            /* set pointer to trait's grid */
    } while (ctrat != trat->head);      /* repeat for all traits */
    gets(line);                 /* clear newline character */
}               /* end ch_elem_cells (celem, item_num) */


/*************************************************>void ch_trat_cells (ctrat)
* FUNCT : change the cell values for all traits of an element
* PARMS : ctrat <-> pointer to the trait that is to be changed
* VARS  : elem -> pointer to the list of elements
* NOTES : ??
*  1) - SPACING : no extra lines output!
*/
void ch_trat_cells (ctrat)
TITLE *ctrat;
{
TITLE *celem;
CELL *ccell;
char line[30];
int number;

  celem = elem->head->next; /* current elem is first elem of elem list */
  ccell = ctrat->grid;
  printf("\n\nPlease rate each of the following %ss on a scale of 1 to ",
      elem->type);
  printf("5, where\n     (%s) is 5 and (%s) is 1.  (type any negative",
      ctrat->name, ctrat->name2);
  printf("\n     number to exit, or '0' to remain unchanged)\n\n");
  do {              /*    do for all elements    */
      printf("Value for %s is currently %d, new value is -> ",
          celem->name, ccell->value);
      scanf("%d",&number);
      if (number < 0) {
          gets(line);
          printf("\n\n\n");
          return;
        }     /* end if */
      if ((number < 0) || (number > 5)) {
          printf("     Please enter a number between 1 and 5.\n");
          continue;
        }     /* end if */
      if ((number == 0) && (ccell->value == 0)) {
          printf("     A number between 1 and 5 must be entered.\n");
          continue;
        }     /* end if */
      if (number != 0) ccell->value = number;
      celem = celem->next;              /* point to next element */
      ccell = ccell->next;              /* point to next cell */
  } while (celem != elem->head);        /* repeat for all elements */
  gets(line);               /* clear newline character */
}               /* end ch_trat_cells (ctrat) */


/*************************************************>void add_elem_cells (item_num)
* FUNCT : add all of the cells for the element number passed
* PARMS : item_num -> position which the new cells should be added to trat list
* VARS  : trat <-> pointer to the trait list
* NOTES : ??
*  1) - There needs to be a trait grid present to add the grid values for
*         an additional element BUT no explicit checking is done
*/
void add_elem_cells (item_num)
int item_num;
{
TITLE *ctrat;
CELL *ccell, *newcell;
int index;

  ctrat = trat->head->next; /* current trat is the first trat in list */
  ccell = ctrat->grid;
  do {              /* do for all traits */
      for (index = 1;index < item_num; index++) /* move to position where */
          ccell = ccell->next;                  /*  the new cell goes */
      newcell = (CELL *)malloc(sizeof(CELL));
      if (item_num == 1)  /* if it is to be the first cell of grid */
          ctrat->grid = newcell;        /* re-assign grid pointer */
      newcell->next = ccell;            /* Attach new node to list */
      newcell->last = ccell->last;
      ccell->last->next = newcell;
      ccell->last = newcell;
      newcell->value = 0;               /* initialize value in new cell */
      ctrat = ctrat->next;              /* set pointer to next trait */
      ccell = ctrat->grid;              /* set pointer to trait's grid */
    } while (ctrat != trat->head);          /* repeat for all traits */
}               /* end add_elem_cells (item_num) */


/*************************************************>void add_trat_cells (ctrat)
* FUNCT : add all of the cells for a newly entered trait
* PARMS : ctrat <-> pointer to a trait just added to the trait list
* VARS  : elem <-> pointer to the element list
* NOTES : ??
*/
void add_trat_cells (ctrat)
TITLE *ctrat;
{
TITLE *celem;
CELL *ccell, *newcell;

  celem = elem->head->next; /* current elem is first elem of elem list */
  ccell = ctrat->grid;
  do {              /*    do for all elements    */
      newcell = (CELL *)malloc(sizeof(CELL));
      if (ccell == NULL) {          /* if no grid, add first cell - The*/
          ccell = newcell;          /*  first time through, both new- */
          ccell->next = ccell;      /*  cell and ccell are equal. This */
          ccell->last = ccell;      /*  block sets up the pointers */
          ctrat->grid = ccell;  }
      newcell->next = ccell->next;      /* Attach new node to list */
      newcell->last = ccell;
      ccell->next = newcell;
      ctrat->grid->last = newcell;
      newcell->value = 0;           /* initialize cell value */
      celem = celem->next;          /* point to next element */
      ccell = newcell;              /* move current cell to head (newcell) */
  } while (celem != elem->head);            /* repeat for all elements */
}               /* end add_trat_cells (ctrat) */


/*************************************************************************/

#include "GRIDEFS6.H"

extern ITEM *elem, *trat;
extern MATCH *m_list;
extern int disp_r; disp_c;

/*              -=<* Directory of Procedures - File ANALY6.C *>=-
*
```

```
********>void analyze_grid()
* FUNCT : do all the analysis of the grid
*
********>void save_all()
* FUNCT : save the element list, the trait list, and/or the grid
*
********>void anal_egrid()
* FUNCT : display, edit, and analyze an element and all its trait values
*
********>void anal_tgrid()
* FUNCT : display, edit, and analyze a trait and all the elements values
*
********>void corr_grid_head (ltem, lines)
* FUNCT : to print out the heading for the correlation matrix
*
********>void ecorr_grid()
* FUNCT : to calculate and print the correlation matrix between the elements
*
********>void tcorr_grid()
* FUNCT : to calculate and print the correlation matrix between the traits
*
********>void ecluster_grid()
* FUNCT : to perform and report the cluster analysis on the elements
*
********>void tcluster_grid()
* FUNCT : to perform and report the cluster analysis on the traits
*
********>void add_match_list(dbl_item, item_a, item_b, score)
* FUNCT : to add a match to the FRONT of match list of items sent to it
*
********>void del_match_list (dbl_item)
* FUNCT : to delete the match list of items sent to it
*
********>void del_all_matches (void)
* FUNCT : to delete all of the match lists
*
********>void report_ematch (dbl_item)
* FUNCT : report the list of matching elements to the user
*
********>void report_tmatch (dbl item)
* FUNCT : report the list of matching traits to the user
*
********>void ch_match_value (void)
* FUNCT : to change the value at which matches are detected
*
*/


/************************************************************>void analyze_grid()
* FUNCT : do all the analysis of the grid
* VARS  : trat -> pointer to the trait and grid list
*         elem -> pointer to the element list
* NOTES : ??
*  1) - This procedure assumes that there is a grid present when it is called
*/
void analyze_grid()
{
char line[30], cmd_ary[MAX_CMDS][10];
int num_cmds, cmd_num, rtn;

  make_cmd_list
          ("CH-MATCH E-ANAL T-ANAL RATE-T RULES AUTO SAVE-ALL LIST DONE HELP",
      &num_cmds, cmd_ary);
  do {
    printf("Your choices for analyzing the present grid are as follows:\n\n");
    help_cmd_list("value", num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {
       /* "CH-MATCH E-ANAL T-ANAL RATE-T RULES AUTO SAVE-ALL LIST DONE HELP" */
      case 0 :        /* no commands match with users input */
              printf("The command you entered, %s, is not understood.",
                  line);
              printf("  Please enter\n             your choice again.\n\n\n\n\n");
              break;
      case 1 : ch_match_value();
              break;
      case 2 : anal_egrid();
              break;
      case 3 : anal_tgrid();
              break;
      case 4 : rate_traits();
              break;
      case 5 : rule_gen();
              break;
      case 6 : rtn=auto_anal();
              printf("Returned with a value of %d.\n", rtn);
              break;
      case 7 : save_all();
              break;
      case 8 : list_grid();
              break;
      case 9 : break;/* NOTE if this num changes, change DO WHILE condition */
      case 10 : break;  /* HELP - the command list will print automatically */
     default : printf("ERROR - analysis command not accounted for!!!");
              gets(line);
              printf("\n\n\n\n\n");
      }       /* end switch */
  } while (cmd_num != 9);     /* end do-while */
}             /* end analyze_grid() */


/************************************************************>void save_all()
* FUNCT : save the element list, the trait list, and/or the grid
* VARS  : trat <-> pointer to the trait list
*         elem <-> pointer to the element list
* NOTES : ??
*/
void save_all()
{
char line[30], cmd_ary[MAX_CMDS][10];
int num_cmds, cmd_num;

  make_cmd_list ("SAVE E-SAVE T-SAVE LIST LIST-E LIST-T DONE HELP",
      &num_cmds, cmd_ary);
  do {
    printf("Your choices for saving the current lists are as follows:\n\n");
    help_cmd_list("value", num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) { /* "SAVE E-SAVE T-SAVE LIST LIST-E LIST-T DONE HELP" */
      case 0 :        /* no commands match with users input */
              printf("The command you entered, %s, is not understood.",
                  line);
              printf("  Please enter\n             your choice again.\n\n\n\n\n");
              break;
      case 1 : save_item_list (trat, 0);
              break;
      case 2 : save_item_list (elem, 0);
              break;
      case 3 : save_item_list (trat, 1);  /* save only trait list */
              break;
      case 4 : list_grid();
              break;
```

```
      case 5 : list_item(elem);
               break;
      case 6 : list_item(trat);
               break;
      case 7 : break;/* NOTE if this num changes, change DO-WHILE condition */
      case 8 : break;  /* HELP - the command list will print automatically */
     default : printf("ERROR - analysis command not accounted for!!!");
               gets(line);
               printf("\n\n\n\n");
      }        /* end switch */
  } while (cmd_num != 7);      /* end do-while */
}                  /* end save_all() */


/*************************************************************>void anal_egrid()
* FUNCT : display, edit, and analyze an element and all its trait values
* VARS  : trat <-> pointer to the trait list
*         elem <-> pointer to the element list
* NOTES : ??
*/
void anal_egrid()
{
char line[30], cmd_ary[MAX_CMDS][10];
int num_cmds, cmd_num;

  make_cmd_list
      ("E-CORR E-CLUST RPT-EMAT EDIT-E EDIT-T LIST-E LIST-G DONE HELP",
      &num_cmds, cmd_ary);
  do {
    printf("Your choices for editing the grid elements are as follows:\n\n");
    help_cmd_list("value", num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {
       /**E CORR E-CLUST RPT-EMAT EDIT-E EDIT-T LIST-E LIST-G DONE HELP**/
      case 0 :          /* no commands match with users input */
               printf("The command you entered, %s, is not understood.",
                   line);
               printf("  Please enter\n             your choice again.\n\n\n\n");
               break;
      case 1 : ecorr_grid();
               if (m_list->dbl_e != NULL) {          /* if matches were found */
                   printf("NOTE - Correlation");
                   report_ematch (m_list->dbl_e); }  /*   inform user */
               break;
      case 2 : ecluster_grid();
               if (m_list->clstr_e != NULL) {        /* if matches were found */
                   printf("NOTE - Cluster");
                   report_ematch (m_list->clstr_e); }  /*   inform user */
               break;
      case 3 : if (m_list->dbl_e != NULL) {
                   printf("NOTE - Correlation");
                   report_ematch (m_list->dbl_e); }
               if (m_list->clstr_e != NULL) {
                   printf("NOTE - Cluster");
                   report_ematch (m_list->clstr_e); }
               break;
      case 4 : edit_item_list (elem);
               break;
      case 5 : edit_item_list (trat);
               break;
      case 6 : list_item (elem);
               break;
      case 7 : list_grid();
               break;
      case 8 : break; /* NOTE if this num changes, change DO-WHILE condition */
      case 9 : break;  /* HELP - the command list will print automatically */
     default : printf("ERROR - analysis command not accounted for!!!");
               gets(line);
               printf("\n\n\n\n");
      }        /* end switch */
  } while (cmd_num != 8);      /* end do-while */
}         /* end anal_egrid() */


/*************************************************************>void anal_tgrid()
* FUNCT : display, edit, and analyze a trait and all the elements values
* VARS  : trat <-> pointer to the trait list
*         elem <-> pointer to the element list
* NOTES : ??
*/
void anal_tgrid()
{
char line[30], cmd_ary[MAX_CMDS][10];
int num_cmds, cmd_num;

  make_cmd_list
      ("T-CORR T-CLUST RPT-TMAT EDIT-T EDIT-E LIST-T LIST-G DONE HELP",
      &num_cmds, cmd_ary);
  do {
    printf("Your choices for editing the grid traits are as follows:\n\n");
    help_cmd_list("value", num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {
       /**T-CORR T-CLUST RPT-TMAT EDIT-T EDIT E LIST-T LIST-G DONE HELP**/
      case 0 :          /* no commands match with users input */
               printf("The command you entered, %s, is not understood.",
                   line);
               printf("  Please enter\n             your choice again.\n\n\n\n");
               break;
      case 1 : tcorr_grid();
               if (m_list->dbl_t != NULL) {          /* if matches were found */
                   printf("NOTE - Correlation");
                   report_tmatch (m_list->dbl_t); }    /*   inform user */
               break;
      case 2 : tcluster_grid();
               if (m_list->clstr_t != NULL) {        /* if matches were found */
                   printf("NOTE - Cluster");
                   report_tmatch (m_list->clstr_t); }  /*   inform user */
               break;
      case 3 : if (m_list->dbl_t != NULL) {
                   printf("NOTE - Correlation");
                   report_tmatch (m_list->dbl_t); }
               if (m_list->clstr_t != NULL) {
                   printf("NOTE - Cluster");
                   report_tmatch (m_list->clstr_t); }
               break;
      case 4 : edit_item_list (trat);
               break;
      case 5 : edit_item_list (elem);
               break;
      case 6 : list_item (trat);
               break;
      case 7 : list_grid();
               break;
      case 8 : break; /* NOTE if this num changes, change DO-WHILE condition */
      case 9 : break;  /* HELP - the command list will print automatically */
     default : printf("ERROR - analysis command not accounted for!!!");
               gets(line);
               printf("\n\n\n\n");
      }        /* end switch */
  } while (cmd_num != 8);      /* end do-while */
}         /* end anal_tgrid() */
```

```
/*****************************************>void corr_grid_head (item, lines)
* FUNCT : to print out the heading for the correlation matrix
* PARMS : item -> pointer to the item list
*         &lines <-> a pointer to the number of lines output to the screen
* VARS  : disp_c -> the number of columns the current display uses
* NOTES : ??
*/
void corr_grid_head (item, lines)
ITEM *item;
int *lines;
{
int index, position, max_position;

  max_position = (disp_c - 24) / 4; /* 22thame+ 2DESQ border = 24, 4 per num */
     /* print out titles */
  printf("%13s         ", item->type);
  position = 0;
  for (index = 1;index <= item->number;index++) {
      if (position >= max_position) {
          printf("\n                              ");  /* space over 24, 2 spaces  */
          (*lines)++;                                  /* to right of values above */
          position = 0;
          max_position--;/* one less position to use because of spacing over */
        }     /* end if */
      printf(" %2d ", index);
      position++;
    }     /* end for */
     /* print out line seperating title from correlations */
  max_position = (disp_c - 24) / 4; /* note it could have been changed above */
  position = 0;
  printf("\n ====================");
  (*lines)++;
  for (index = 1;index <= item->number;index++) {
      if (position >= max_position) break;       /* only need one row of '=' */
      printf("====");
      position++;
    }     /* end for */
  printf("\n");
  (*lines)++;
}             /* end corr_grid_head (item, lines) */


/*************************************************************>void ecorr_grid()
* FUNCT : to calculate and print the correlation matrix between the elements
* VARS  : trat -> pointer to the trait and grid list
*         elem -> pointer to the element list
*         m_list <-> pointer to the match list
* NOTES : ??
*  1) - this procedure assumes that there is a grid present - see loop
*       conditions on the while loops
*/
void ecorr_grid()
{
TITLE *elema, *elemb, *ctrat;
CELL *cella, *cellb;
char line[30];
int elema_num, elemb_num, index, corr, totcorr, lines_out, position,
      max_position, score;

  if (trat->head->next->grid == NULL) {   /* if the grid doesn't exist */
      printf("The grid does not yet exist.  The grid must be entered before ");
      printf("a correlation\n   between elements can be calculated.\n\n\n\n");
      return;                             /*  print out a message and return */
    }         /* end if */
  totcorr = trat->number * 2;  /* compute the maximum correlation allowed */
  del_match_list (&m_list->dbl_e);  /*  erase any close matches found before */
  lines_out = 5;  /* note: 2 for description, 3 for 'hit rtn to continue'*/
  printf("Below are listed the correlations between the elements:\n\n");
  corr_grid_head (elem, &lines_out);
  elema = elem->head->next;
  elema_num = 1;       /* element number to do correlation on */
  while (elema != elem->head) {        /* do correlation for all elements */
      max_position = (disp_c - 24) / 4; /* reset m_p for each line of output */
                    /* 2DESQ border +4 num +16 title +2sp = 24, 4 per num */
      if (lines_out >= disp_r) {
          printf("\nPlease hit RETURN to continue, or type QUIT to ");
          printf("stop listing - ");
          gets(line);
          strupr(line);
          if (strstr(line, "QUIT") != NULL) {
              printf("\n\n\n\n\n");
              return;  }
          printf("\n");
          lines_out = 5;
          corr_grid_head (elem, &lines_out);
        } /* end if */
      elemb = elema->next;
      elemb_num = elema_num + 1;  /* elem num to compare against elema */
      position = 0;
      printf(" %2d. %15s  ", elema_num, elema->name); /* print elem name */
      for (index = 1;index < elema_num;index++){/* space to correct position */
          if (position >= max_position) {
              printf("\n                              ");/*space over 24, 2 spaces*/
              lines_out++;                              /*to right of values above*/
              position = 0;
              max_position--;/*one less position to use because spacing over*/
            }         /* end if */
          printf("    ");
          position++;
        };        /* end for */
      if (position >= max_position) { /*impt when printing ' -- ' in last col*/
          printf("\n                              ");/*space over 24, 2 spaces*/
          lines_out++;                              /*to right of values above*/
          position = 0;
          max_position--;/*one less position to use because spacing over*/
        }         /* end if */
      printf(" -- ");     /* print this when row and col nums are the same */
      position++;
      while (elemb != elem->head) {   /* compare against all other elements */
          corr = 0;
          ctrat = trat->head->next;
          while (ctrat != trat->head) {   /* go through all traits to */
              cella = ctrat->grid;        /* calculate the correlation */
              for (index = 1;index < elema_num; index++) cella = cella->next;
              cellb = ctrat->grid;
              for (index = 1;index < elemb_num; index++) cellb = cellb->next;
              if (cella->value == cellb->value) corr = corr + 2; /* direct */
              if ((cella->value + cellb->value == 3) ||   /* partial match */
                  (cella->value + cellb->value == 9)) corr++;
              ctrat = ctrat->next;
            }     /* end while */
          if (position >= max_position) {
              printf("\n                              ");/*space over 24, 2 spaces*/
              lines_out++;                              /*to right of values above*/
              position = 0;
              max_position--;/*one less position to use because spacing over*/
            }         /* end if */
          printf(" %2d ", corr);
          score = ((corr * 100)/(totcorr));    /* this is done to preserve 2 */
                /* decimal places in int result - same as (corr/tcorr)*100=% */
          if (score >= m_list->level)     /* if a match occures, add to list */
              add_match_list (&m_list->dbl_e, elema_num, elemb_num, score);
          position++;
          elemb_num++;          /* compare against next element */
          elemb = elemb->next;
        }     /* end while */
      printf("\n");
```

```
      lines_out++;
      elema_num++;            /* do correlation for next element */
      elema = elema->next;
    }     /* end while */
  printf("\nThe maximum correlation between any two elements is %d.\n\n",
      totcorr);
  printf("Please hit RETURN to continue -");
  gets(line);
  printf("\n\n\n\n");
}             /* end ecorr_grid() */


/*************************************************************>void tcorr_grid()
* FUNCT : to calculate and print the correlation matrix between the traits
* VARS  : trat -> pointer to the trait and grid list
*         m_list <-> pointer to the match list
* NOTES : ??
*  1) - this procedure assumes that there is a grid present - see loop
*       conditions on the while loops
*/
void tcorr_grid()
{
TITLE *trata, *tratb;
CELL *cella, *cellb;
char line[30];
int index, trata_num, tratb_num, corr, totcorr, lines_out, position,
      max_position, score;

  if (trat->head->next->grid == NULL) {   /* if the grid dosn't exist */
      printf("The grid does not yet exist.  The grid must be entered before ");
      printf("a correlation\n   between traits can be calculated.\n\n\n\n");
      return;                             /*  print out a message and return */
    }         /* end if */
  totcorr = elem->number * 2;  /* compute the maximum correlation allowed */
  del_match_list (&m_list->dbl_t);   /* erase any close matches found before */
  lines_out = 5;  /* note: 2 for description, 3 for 'hit rtn to continue'*/
  printf("Below are listed the correlations between the traits:\n\n");
  corr_grid_head (trat, &lines_out);
  trata = trat->head->next;
  trata_num = 1;        /* trait num for doing correlation on */
  while (trata != trat->head) {        /* do correlation for all traits */
      max_position = (disp_c - 24) / 4; /* reset m_p for each line of output */
                     /* 2DESQ border +4 num +16 title +2sp = 24, 4 per num */
      if (lines_out >= disp_r) {
          printf("\nPlease hit RETURN to continue, or type QUIT to ");
          printf("stop listing - ");
          gets(line);
          strupr(line);
          if (strstr(line, "QUIT") != NULL) {
              printf("\n\n\n\n");
              return;  }
          printf("\n");
          lines_out = 5;
          corr_grid_head (trat, &lines_out);
        } /* end if */
      tratb = trata->next;
      tratb_num = trata_num + 1;
      position = 0;
      printf(" %2d. %15s  ", trata_num, trata->name); /* print trait name */
      for (index = 1;index < trata_num;index++) {/* space to correct position */
          if (position >= max_position) {
              printf("\n                              ");/*space over 24, 2 spaces*/
              lines_out++;                    /*to right of values above*/
              position = 0;
              max_position--;/*one less position to use because spacing over*/
            }         /* end if */
          printf("    ");
          position++;
        };        /* end for */
      if (position >= max_position) { /*impt when printing ' -- ' in last col*/
          printf("\n                              ");/*space over 24, 2 spaces*/
          lines_out++;                    /*to right of values above*/
          position = 0;
          max_position--;/*one less position to use because spacing over*/
        }         /* end if */
      printf(" -- ");     /* print this when row and col nums are the same */
      position++;
      while (tratb != trat->head) {       /* compare against all other traits */
          corr = 0;
          cella = trata->grid;
          cellb = tratb->grid;
          do {              /* calculate the correlation */ /* direct match */
              if (cella->value == cellb->value) corr = corr + 2;
              if ((cella->value + cellb->value == 3) ||   /* partial match */
                  (cella->value + cellb->value == 9)) corr++;
              cella = cella->next;
              cellb = cellb->next;
            } while (cella != trata->grid);
          if (position >= max_position) {
              printf("\n                              ");/*space over 24, 2 spaces*/
              lines_out++;                    /*to right of values above*/
              position = 0;
              max_position--;/*one less position to use because spacing over*/
            }         /* end if */
          printf(" %2d ", corr);
          score = ((corr * 100) / (totcorr));
           /* this is done to preserve 2 decimal places in the intger result */
          if (score >= m_list->level)     /* if a match occures, add to list */
              add_match_list (&m_list->dbl_t, trata_num, tratb_num, score);
          position++;
          tratb_num++;
          tratb = tratb->next;
        }         /* end while */
      printf("\n");
      lines_out++;
      trata_num++;      /* do correlation for next trait */
      trata = trata->next;
    }     /* end while */
  printf("\nThe maximum correlation between any two traits is %d.\n\n",
      totcorr);
  printf("Please hit RETURN to continue -");
  gets(line);
  printf("\n\n\n\n");
}             /* end tcorr_grid() */


/*************************************************************>void ecluster_grid()
* FUNCT : to perform and report the cluster analysis on the elements
* VARS  : trat -> pointer to the trait and grid list
*         elem -> pointer to the element list
*         m_list <-> pointer to the match list
* NOTES : ??
*  1) - this procedure assumes that there is a grid present - see loop
*       conditions on the while loops
*/
void ecluster_grid()
{
TITLE *elema, *elemb, *ctrat;
CELL *cella, *cellb;
char line[30];
int elema_num, elemb_num, index, corr, lines_out, position,
      max_position, score;

  if (trat->head->next->grid == NULL) {   /* if the grid dosn't exist */
      printf("The grid does not yet exist.  The grid must be entered before ");
      printf("a correlation\n   between elements can be calculated.\n\n\n\n");
      return;                             /*  print out a message and return */
    }         /* end if */
```

```
del_match_list (&m_list->clstr_e);  /*  erase any close matches found before */
lines_out = 5;  /* note: 2 for description, 3 for 'hit rtn to continue'*/
printf("Below are listed the cluster analysis results of the elements:\n\n");
corr_grid_head (elem, &lines_out);
elema = elem->head->next;
elema_num = 1;       /* element number to do correlation on */
while (elema != elem->head) {        /* do correlation for all elements */
    max_position = (disp_c - 24) / 4; /* reset m_p for each line of output */
                   /* 2DESQ border +4 num +16 title +2sp = 24, 4 per num */
    if (lines_out >= disp_r) {
        printf("\nPlease hit RETURN to continue, or type QUIT to ");
        printf("stop listing - ");
        gets(line);
        strupr(line);
        if (strstr(line, "QUIT") != NULL) {
            printf("\n\n\n\n");
            return;  }
        printf("\n");
        lines_out = 5;
        corr_grid_head (elem, &lines_out);
      } /* end if */
    elemb = elema->next;
    elemb_num = elema_num + 1;  /* elem num to compare against elema */
    position = 0;
    printf(" %2d. %15s  ", elema_num, elema->name); /* print elem name */
    for (index = 1;index < elema_num;index++){/* space to correct position */
        if (position >= max_position) {
            printf("\n                                ");/*space over 24, 2 spaces*/
            lines_out++;                      /*to right of values above*/
            position = 0;
            max_position--;/*one less position to use because spacing over*/
          }         /* end if */
        printf("    ");
        position++;
      };          /* end for */
    if (position >= max_position) { /*impt when printing ' -- ' in last col*/
        printf("\n                              ");/*space over 24, 2 spaces*/
        lines_out++;                          /*to right of values above*/
        position = 0;
        max_position--;/*one less position to use because spacing over*/
      }           /* end if */
    printf(" -- ");      /* print this when row and col nums are the same */
    position++;
    while (elemb != elem->head) {   /* compare against all other elements */
        corr = 0;
        ctrat = trat->head->next;
        while (ctrat != trat->head) {   /* go through all traits to */
            cella = ctrat->grid;        /* calculate the correlation */
            for (index = 1;index < elema_num; index++) cella = cella->next;
            cellb = ctrat->grid;
            for (index = 1;index < elemb_num; index++) cellb = cellb->next;
            corr = corr + abs(cella->value - cellb->value);
            ctrat = ctrat->next;
          }      /* end while */
        if (position >= max_position) {
            printf("\n                                ");/*space over 24, 2 spaces*/
            lines_out++;                      /*to right of values above*/
            position = 0;
            max_position--;/*one less position to use because spacing over*/
          }         /* end if */
        score = (-100 * corr)/((5 - 1) * trat->number) + 100;
        printf("%3d ", score);
        if (score >= m_list->level)/* if a match occures, add to list */
            add_match_list (&m_list->clstr_e, elema_num, elemb_num, score);
        position++;
        elemb_num++;         /* compare against next element */
        elemb = elemb->next;
      }      /* end while */
    printf("\n");
    lines_out++;
    elema_num++;             /* do correlation for next element */
    elema = elema->next;
  }      /* end while */
printf("\nPlease hit RETURN to continue -");
gets(line);
printf("\n\n\n\n\n");
}              /* end cclustar_grid() */


/*******************************************************>void tcluster_grid()
* FUNCT : to perform and report the cluster analysis on the traits
* VARS  : trat -> pointer to the trait and grid list
*         m_list <-> pointer to the match list
* NOTES : ??
*  1) - this procedure assumes that there is a grid present - see loop
*       conditions on the while loops
*/
void tcluster_grid()
{
TITLE *trata, *tratb;
CELL *cella, *cellb;
char line[30];
int index, trata_num, tratb_num, corr, lines_out, position,
      max_position, score;

  if (trat->head->next->grid == NULL) {   /* if the grid dosn't exist */
      printf("The grid does not yet exist.  The grid must be entered before ");
      printf("a correlation\n   between traits can be calculated.\n\n\n\n");
      return;                             /*  print out a message and return */
    }         /* end if */
  del_match_list (&m_list->clstr_t);   /* erase any close matches found before */
  lines_out = 7;  /* note: 4 for description, 3 for 'hit rtn to continue'*/
  printf("Below are listed the cluster analysis results of the traits:\n");
  printf("  (Negative values indicate that traits are opposites.)\n\n");
  corr_grid_head (trat, &lines_out);
  trata = trat->head->next;
  trata_num = 1;       /* trait num for doing correlation on */
  while (trata != trat->head) {       /* do correlation for all traits */
      max_position = (disp_c - 24) / 4; /* reset m_p for each line of output */
                     /* 2DESQ border +4 num +16 title +2sp = 24, 4 per num */
      if (lines_out >= disp_r) {          /* new page */
          printf("\nPlease hit RETURN to continue, or type QUIT to ");
          printf("stop listing - ");
          gets(line);
          strupr(line);
          if (strstr(line, "QUIT") != NULL) {
              printf("\n\n\n\n");
              return;  }
          printf("\n");
          lines_out = 5;
          corr_grid_head (trat, &lines_out);
        } /* end if */
      tratb = trat->head->next;
      tratb_num = 1;
      position = 0;
      printf(" %2d. %15s ", trata_num, trata->name); /* print trait name */
      while (tratb != trat->head) {      /* compare against all other traits */
          if (position >= max_position) {
              printf("\n                              ");/*space over 23, 2 spaces*/
              lines_out++;                      /*to right of values above*/
              position = 0;
              max_position--;/*one less position to use because spacing over*/
            }         /* end if */
          corr = 0;
          cella = trata->grid;
/*1*/     if (trata_num == tratb_num) {  /* tratA&B are same, no calculation */
              printf("  --");                     /* print this when row and */
              position++;                       /*   col nums are the same */
```

```
        }  /* end if I */
        else { /* else I */
    /*J*/   if (trata_num < tratb_num) {          /* trataA<B, use normal B */
            cellb = tratb->grid;
            do {
                corr = corr + abs(cella->value - cellb->value);
                cella = cella->next;
                cellb = cellb->next;
              } while (cella != trata->grid);
            }     /* end if J */
            else  /* else J */
      /*K*/   if (trata_num > tratb_num) {  /* trataA>B, use reverse of B */
                tratb_num = tratb_num * -1;  /* use -B to denote reverse */
                cellb = tratb->grid->last;          /* start at end of B */
                do {
                    corr = corr + abs(cella->value - cellb->value);
                    cella = cella->next;
                    cellb = cellb->last;
                  } while (cella != trata->grid);
                }  /* end if J and if-else J */
            score = (-200 * corr)/((5 - 1) * elem->number) + 100;
            printf("%4d", score);
            if (abs(score) >= m_list->level) /*if match occures,add to list*/
              add_match_list (&m_list->clstr_t, trata_num, tratb_num, score);
            tratb_num = abs(tratb_num);     /* if -B was used, reset it for */
            position++;                          /* the next iteration */
          }  /* end if-else I */
        tratb_num++;
        tratb = tratb->next;
      }     /* end while */
      printf("\n");
      lines_out++;
      trata_num++;      /* do correlation for next trait */
      trata = trata->next;
    }     /* end while */
  printf("\nNote the numbers in the lower left-hand side of the table result");
  printf(" from\n  inverting one of the trait poles before doing the ");
printf("analysis.\n\nPlease hit RETURN to continue -");
  gets(line);
  printf("\n\n\n\n");
}               /* end tcluster_grid() */


/************************>void add_match_list (dbl_item, item_a, item_b, score)
* FUNCT : to add a match to the FRONT of match list of items sent to it
* VARS  : dbl_item <-> a pointer to the pointer to the item match list
*         item-a -> the number of matching item 'a'
*         item-b -> the number of matching item 'b'
*         score -> their score, or percent of matching (0-100)
* NOTES : ??
*/
void add_match_list (dbl_item, item_a, item_b, score)
DOUBLE **dbl_item;
int item_a, item_b, score;
{
DOUBLE *new_dbl, *curnt_dbl;

  new_dbl = (DOUBLE *)malloc(sizeof(DOUBLE)); /* create new node */
  curnt_dbl = *dbl_item; /*Note: use bubble sort-places largest matches first*/
  if ( *dbl_item == NULL || abs(curnt_dbl->score) < abs(score)) {
      new_dbl->next = *dbl_item;    /*if new score is */
      *dbl_item = new_dbl;          /*the first OR is larger then first score,*/
    }                               /* put it first on the singly linked list */
    else {
      while ((abs(curnt_dbl->next->score) > abs(score)) &&
        (curnt_dbl->next != NULL)) /* else find where */
          curnt_dbl = curnt_dbl->next;                  /*    it belongs */
      new_dbl->next = curnt_dbl->next;          /* and insert it before other */
      curnt_dbl->next = new_dbl;                /* equal or lesser scores */
      }     /* end if else */
  new_dbl->item_a = item_a;                     /* and assign the values */
  new_dbl->item_b = item_b;
  new_dbl->score = score;
}               /* end add_match_list (dbl_item, item_a, item_b, score) */


/************************************************>void del_match_list (dbl_item)
* FUNCT : to delete the match list of items sent to it
* VARS  : dbl_item <- a pointer to the pointer to the item match list
* NOTES : ??
*/
void del_match_list (dbl_item)
DOUBLE **dbl_item;
{
DOUBLE *current;
int index;

  while (*dbl_item != NULL) {
      current = *dbl_item;   /* point to rest of list */
      *dbl_item = current->next;      /* re-assign dbl_item to point to list */
      free(current);              /* remove FIRST node of list and delete it */
    }               /* end while */
}                   /* end del_match_list (dbl_item) */


/************************************************>void del_all_matches (void)
* FUNCT : to delete all of the match lists
* NOTES : ??
*/
void del_all_matches()
{
  del_match_list (&m_list->dbl_e);
  del_match_list (&m_list->dbl_t);
  del_match_list (&m_list->clstr_e);
  del_match_list (&m_list->clstr_t);
}                   /* end del_all_matches() */


/************************************************>void report_ematch (dbl_item)
* FUNCT : report the list of matching elements to the user
* NOTES : ??
*  1) - The type of analysis must be printed out before this function is called
*         EX: "NOTE - Correlation"  or "NOTE - Cluster"
*/
void report_ematch (dbl_item)
DOUBLE *dbl_item;
{
DOUBLE *m_item;
char line[20];
int lines_out;

  lines_out = 7; /* 5 for text & spaces, 2 for 'Please hit...' */
  printf(" analysis has determined that several of the elements in\n  the ");
  printf("grid have similar values for at least %d percent of the traits\n",
      m_list->level);
  printf("  currently present in the grid.  The similar elements are listed");
  printf(" below:\n\n");
  m_item = dbl_item;
  while (m_item != NULL) {
      if (lines_out >= disp_r) {
          printf("\nPlease hit RETURN to continue -");
          gets(line);
          printf("\n\n\n");
          lines_out = 2;
        }  /* end if */
      printf(" A %3d%% match : elements %2d {%s} and %2d {%s}.\n",
          m_item->score, m_item->item_a, find_name(elem, m_item->item_a),
          m_item->item_b, find_name(elem, m_item->item_b));
```

```
      lines_out++;
      m_item = m_item->next;
    }           /* end while */
  printf("\nPlease hit RETURN to continue -");
  gets(line);
  printf("\n\n\n\nMatching elements indicates that one of two conditions ");
  printf("exists:\n\n");
  printf(" -the first indicates that the problem at hand has not been\n");
  printf("  completely specified, and that another trait needs to be\n");
  printf("  entered in order to distinguish between those two elements.\n\n");
  printf(" -the second represents the possibility that the two elements are\n");
  printf("  possibly related in some fashion.  The solution to this is to\n");
  printf("  create a new element that represents the two matching elements.\n");
  printf("\n\nPlease hit RETURN to continue -");
  gets(line);
  printf("\n\n\n\n");
}             /* end report_ematch (dbl_item) */


/*************************************************>void report_tmatch (dbl_item)
* FUNCT : report the list of matching traits to the user
* NOTES : ??
*  1) - The type of analysis must be printed out before this function is called
*         EX: "NOTE - Correlation"  or "NOTE - Cluster"
*/
void report_tmatch (dbl_item)
DOUBLE *dbl_item;
{
DOUBLE *m_item;
char line[20], note[12];
int lines_out;

  lines_out = 7; /* 5 for text & spaces, 2 for 'Please hit...' */
  printf(" analysis has determined that several of the traits in\n  the ");
  printf("grid have similar values for at least %d percent of the elements\n",
      m_list->level);
  printf("  currently present in the grid.  The similar traits are listed ");
  printf("below\n  (Negative values indicate that traits are opposites):\n\n");
  m_item = dbl_item;
  while (m_item != NULL) {
      if (lines_out >= disp_r) {
          printf("\nPlease hit RETURN to continue -");
          gets(line);
          printf("\n\n\n");
          lines_out = 2;
        }  /* end if */
      if (m_item->item_b < 0)
          strcpy(note," reversed ");
        else strcpy(note," ");
      printf(" A %3d%% match : traits %2d {%s} and %2d%s{%s}.\n",
          m_item->score, m_item->item_a, find_name(trat, m_item->item_a),
          abs(m_item->item_b), note, find_name(trat, m_item->item_b));
      lines_out++;
      m_item = m_item->next;
    }           /* end while */
  printf("\nPlease hit RETURN to continue -");
  gets(line);
  printf("\n\n\nMatching traits indicates that the two traits can possibly ");
  printf("be\n  combine into one new trait.\n\n");
  printf("\nPlease hit RETURN to continue -");
  gets(line);
  printf("\n\n\n\n");
}             /* end report_tmatch (dbl_item) */


/*************************************************>void ch_match_value (void)
* FUNCT : to change the value at which matches are detected
* NOTES : ??
*/
```

```
void ch_match_value()
{
char line[20];
int number;

  printf("Currently, elements and traits are reported as matching if they ");
  printf("are\n  over %d percent the same.  Please enter a number between ",
      m_list->level);
  printf("1 and 100 that\n  indicates what percent you would like the ");
  printf("matching value to be?\n  (enter 0 to remain unchanged)\n\n");
  printf("The matching value is currently %d, the new matching value is -> ",
      m_list->level);
  scanf("%d",&number);
  gets(line);
  if ((number < 1) || (number > 100)) {
    printf("\nThe matching value has remained unchanged at %d percent.\n\n\n\n",
      m_list->level);
    return;
  }       /* end if */
  m_list->level = number;
  printf("\nElements and traits will now be reported as matching if they ");
  printf("are\n  over %d percent the same.\n\n\n\n", m_list->level);
  del_all_matches();  /* delete all matches found at previous match value */
}                     /* end ch_match_value() */


/*************************************************>void rate_traits (void)
* FUNCT : rate the importance of each trait in the grid
* VARS  : trat <-> pointer to the trait list
* NOTES : ??
*/
void rate_traits()
{
TITLE *ctrat;
char line[30];
int number;

  printf("Please rate each of the following traits based on their importance");
  printf(" to the\n  problem at hand, using a scale of 1 (least important) ");
  printf("to 5 (most important).\n  (type any negative number to exit, or ");
  printf("'0' to remain unchanged)\n\n");
  ctrat = trat->head->next;
  do {            /*   do for all traits   */
      printf("The importance of %s {%s} is currently %d, new value is -> ",
          trat->type, ctrat->label, ctrat->weight);
      scanf("%d",&number);
      if (number < 0) {
          gets(line);
          printf("\n\n\n");
          return;
        }     /* end if */
      if ((number < 0) || (number > 5)) {
          printf("     Please enter a number between 1 and 5.\n");
          continue;
        }     /* end if */
      if ((number == 0) && (ctrat->weight == 0)) {
          printf("     Each trait must be rated.  Please enter a number ");
          printf("between 1 and 5.\n");
          continue;
        }     /* end if */
      if (number != 0) ctrat->weight = number;
      ctrat = ctrat->next;            /* point to next trait */
  } while (ctrat != trat->head);      /* repeat for all trait */
  gets(line);             /* clear newline character */
  printf("\n\n\n");
}                 /* end rate_traits() */


/*************************************************************************/
```

```
#include "GRIDEFS6.H"

extern ITEM *elem, *trat;
extern MATCH *m_list;
extern int disp_i; disp_c;

/*              -=<* Directory of Procedures - File NEW6.C *>=-
*
********>void rule_gen (void)
* FUNCT : present various forms of rules that can be generated
*
********>void ch2_t_prolog (string)
* FUNCT : change the string to a format recognized by turbo prolog
*
********>void ch2_hc_prolog (string)
* FUNCT : change the string to a format recognized by hc prolog
*
********>int scale_factor (trait_number)
* FUNCT : determine the scaling factor to use based on the number of traits
*
********>void gen_t_prolog (void)
* FUNCT : generat a rule base from the grid for turbo prolog
*
********>void gen_hc_prolog (void)
* FUNCT : generat a rule base from the grid for hc prolog
*
********>void gen_prolog (void)
* FUNCT : generat a rule base w/entailments from the grid for prolog
*
********>void auto_anal (void)
* FUNCT : perform the analysis and generation of the rule base from the grid
*
********>char *find_name (item, num)
* FUNCT : To find the name of the num'th item on the item list.
*
********>char *find_name2 (item, num)
* FUNCT : To find the name2 of the num'th item on the item list.
*
********>char *find_label (item, num)
* FUNCT : To find the label of the num'th item on the item list.
*
********>void combine_elem (dbl)
* FUNCT : combine the two matching elements noted in the match list
*
********>void verify_elem (dbl)
* FUNCT : verify the values entered for the two matching elements
*
********>void verify_trat (dbl)
* FUNCT : verify the values entered for the two matching traits
*
*/


/*********************************************************>void rule_gen (void)
* FUNCT : present various forms of rules that can be generated
* VARS  : NONE
* NOTES : ??
*/
void rule_gen()
{
char line[30], cmd_ary[MAX_CMDS][10];
int num_cmds, cmd_num;

  if (trat->head->next->weight == 0)  { /* if traits arn't rated, return */
      printf("In order to generate rules from the current grid, you must ");
      printf("rate each of\n     the traits according to their relative ");
      printf("importance.\n\n\n");
      return;
    }     /* end if */

  make_cmd_list ("T-PROLOG HC-PROLOG PROLOG LIST DONE HELP", &num_cmds, cmd_ary);
  do {
    printf("Your choices for generating rules from the present grid are as ");
    printf("follows:\n\n");
    help_cmd_list("value", num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {          /* T-PROLOG HC-PROLOG PROLOG LIST DONE HELP */
      case 0 :          /* no commands match with users input */
              printf("The command you entered, %s, is not understood.",
                  line);
              printf("  Please enter\n             your choice again.\n\n\n\n");
              break;
      case 1 : gen_t_prolog();
              break;
      case 2 : gen_hc_prolog();
              break;
      case 3 : gen_prolog();
              break;
      case 4 : list_grid();
              break;
      case 5 : break;/* NOTE if this num changes, change DO-WHILE condition */
      case 6 : break;  /* HELP - the command list will print automatically */
     default : printf("ERROR - analysis command not accounted for!!!");
              gets(line);
              printf("\n\n\n\n");
      }       /* end switch */
  } while (cmd_num != 5);     /* end do-while */
}                 /* end rule_gen() */


/*****************************************************>void ch2_t_prolog (string)
* FUNCT : change the string to a format recognized by turbo prolog
* VARS  : string <-> Pointer to the string to be sanitized
* NOTES : ??
*/
void ch2_t_prolog (string)
char *string;
{
int position;

  strlwr(string);
  position = strcspn(string,"-$=.");
  while (position <= (NAME_LEN - 1)) {
      if (string[position] == NULL) break; /* if no string passed, return */
      string[position] = '_';              /* replace offending char */
      position = strcspn(string,"-$");
    }          /* end while */
}                 /* end ch2_t_prolog (string) */


/*****************************************************>void ch2_hc_prolog (string)
* FUNCT : change the string to a format recognized by hc prolog
* VARS  : string <-> Pointer to the string to be sanitized
* NOTES : ??
*/
void ch2_hc_prolog (string)
char *string;
{
int position;

  strlwr(string);
  position = strcspn(string,"?.()[];:");
  while (position <= (NAME_LEN - 1)) {
      if (string[position] == NULL) break; /* if no string passed, return */
```

```
        string[position] = ' ';              /* replace offending char */
        position = strcspn(string,"?.()[];:");
    }           /* end while */
}                   /* end ch2_hc_prolog (string) */


/*************************************************>int scale_factor (trait_number)
* FUNCT : determine the scaling factor to use based on the number of traits
* VARS  : trait_number -> the current number of traits
* RETNS : returns the scaling factor (times 1000)
* NOTES : ??
*  1) - The number returned MUST be divided by 1000 to obtain the correct
*       scaling factor.
*/
int scale_factor (trait_number)
int trait_number;
{
  switch (trait_number) {
       case 1  : return (999);      case 2  : return (875);
       case 3  : return (750);      case 4  : return (625);
       case 5  : return (500);      case 6  : return (438);
       case 7  : return (375);      case 8  : return (312);
       case 9  : return (297);      case 10 : return (281);
       case 11 : return (250);      case 12 : return (234);
       case 13 : return (219);      case 14 : return (203);
       case 15 : return (188);      case 16 : return (172);
       case 17 : return (164);      case 18 : return (156);
       case 19 : return (148);      case 20 : return (141);
       case 21 : return (137);      case 22 : return (133);
       case 23 : return (125);      case 24 : return (121);
       case 25 : return (117);      case 26 : return (109);
       case 27 : return (105);      case 28 : return (102);
       case 29 : return (100);      case 30 : return (98);
       case 31 : return (94);       case 32 : return (90);
       case 33 : return (88);       case 34 : return (86);
       case 35 : return (84);       case 36 : return (82);
       case 37 : return (78);       case 38 : return (76);
       case 39 : return (74);       case 40 : return (72);
       case 41 : return (71);       case 42 : return (70);
}   /* end switch */
  return (0); /* error condition - should never get this! */
}                   /* end int scale_factor (int) */


/*************************************************>void gen_t_prolog (void)
* FUNCT : generat a rule base from the grid for turbo prolog
* VARS  : NONE
* NOTES : ??
*  1) - This procedure does NOT print out negative numbers, it only
*       inserts a "-" sign before its positive value
*  2) - Note that Turbo-Prolog requires that all strings be in lower case.
*       Because of this, the conversion to lower case is done ONLY on the
*       rules being out put to the rule file.
*/
void gen_t_prolog()
{
TITLE *celem, *ctrat, *trata, *tratb;
CELL *ccell, *cella, *cellb;
FILE *fp, *fp1;
char *flagc, line[50], filename[14], ans;
char name[16], name2[16], label[16], type[16], ename[16];
int AB, AB_, A_B, A_B_, ABN, AB_N, A_BN, A_B_N, scale;
int flag1, pntr, period, index, rule_num, trat_f, grid_f, cert_f;

  printf("Please enter the file name for the Turbo-Prolog rule base.\n");
  strcpy (filename, trat->filename);      /* build rule-base filename */
  period = strcspn(filename,".\0");       /* find period */
  if (filename[period] == '.') {          /* if present, */
      filename[period] = NULL;                /* delete it and extension */
  strcat (filename, ".PRO");              /* add ".PRO" extension */
  }                 /* end IF */
ans = get_filename (filename, ".PRO");
if (ans == 'F') return;
fp = fopen(filename,"r");     /* if file does not exist, fp = NULL */
if (fp != NULL) {   /* If a file already exists, give the user a warning */
    fclose(fp);                       /* first close file */
    printf("The file %s already exists.\n    --NOTE-- continuing will ",
        filename);
    printf("destroy anything currently in that file.\n              Do you ");
    printf("still want to use this file? (YES or *NO) -> ");
    gets(line);
    printf("\n\n");
    strupr(line);
    if (strstr(line,"YES") == NULL) /* if don't want to use the file */
        filename[0] = NULL;       /*  go back and get another filename */
  }               /* end IF */

fp1 = fopen("t-prolog.sav","r");/* open file containing Turbo-Prolog rules */
if (fp1 == NULL) {  /* If file dosn't exist, give user error message */
    printf("NOTE - A working Turbo-Prolog rule base can not be generated\n");
    printf("     since the file 'T-PROLOG.SAV' is not present.  This file\n");
    printf("     contains the inference engine to drive the rules generated\n");
    printf("     by this system.  Please copy this file to the same\n");
    printf("     directory as this program.\n\n");
    printf("  Please hit RETURN to continue - ");
    gets(line);
    printf("\n\n\n\n\n");
    return;
  }                 /* end if */

printf("Rule generation for Turbo-Prolog is about to begin.  Note that since");
printf("there are\n    %d %ss and %d %ss, over %d rules could be generated.",
    elem->number, elem->type, trat->number, trat->type,
    elem->number*trat->number*2);
printf("\nWould you like to proceed?   (*YES or NO) -> ");
gets(line);
printf("\n\n");
strupr(line);
if (strstr(line,"NO") != NULL)          /* if don't want rules generated */
    return;

fp = fopen(filename,"w");
flagc = fgets(line,50,fp1);     /* transfer inference engine */
while (flagc != NULL) {         /*   to beginning of file */
    flag1 = fputs(line,fp);
    flagc = fgets(line,50,fp1);
  }     /* end while */
fclose(fp1);
ctrat = trat->head->next;                               /* print fact rules */
fprintf(fp,"\n/* generated using PCT */\n\n");
fprintf(fp,"ques_lst([\n");
while (ctrat != trat->head) {                   /* do for all traits */
    strcpy (label,ctrat->label); ch2_t_prolog(label); /* change output to */
    strcpy (name,ctrat->name); ch2_t_prolog(name);    /* all lower case as */
    strcpy (name2,ctrat->name2); ch2_t_prolog(name2); /* t-prolog requires */
    fprintf(fp,"     [%s, %s, %s]", label, name, name2);
    ctrat = ctrat->next;        /* move on to next trat */
    if (ctrat != trat->head) fprintf(fp,",\n");  /* do on all but last one */
  }         /* end while */
fprintf(fp,"      ]).\n\n");
fprintf(fp,"maxrules ( %d).\n\n", elem->number*trat->number*2);

ctrat = trat->head->next;                       /* print knowledge rules */
rule_num = 1;
scale = scale_factor (trat->number);            /* get scaling factor */
while (ctrat != trat->head) {                   /* for all traits */
    trat_f = (ctrat->weight * 2);               /* calculate trait factor */
    celem = elem->head->next;
```

```
    ccell = ctrat->grid;
    while (celem != elem->head) {              /* for all elements */
        if ((ccell->value == 5) || (ccell->value == 1))
            grid_f = 10;                       /* assign grid factor */
          else if ((ccell->value == 4) || (ccell->value == 2))
                  grid_f = 5;
                else grid_f = 0;
        cert_f = ((trat_f * grid_f * scale) + 500) / 1000;
              /* calculate certainty factor, range from -100:100 */
     /* cert_f = (cert_f / 2) + 50;     map range from -100:100 to 0:100 */
        if (cert_f != 0) {                     /* output rules */
          strcpy (type,elem->type);        /* change output to all */
          ch2_t_prolog (type);
          strcpy (ename,celem->name);      /*  lower case as turbo- */
          ch2_t_prolog (ename);
          strcpy (label,ctrat->label);     /*  prolog requires */
          ch2_t_prolog (label);
          strcpy (name,ctrat->name);   ch2_t_prolog (name);
          strcpy (name2,ctrat->name2);   ch2_t_prolog (name2);
          if ((ccell->value == 5) || (ccell->value == 4)) {
            fprintf(fp,"rule (%d, %s, %s, %d) :- fact(%s, %s).\n", rule_num++,
              type, ename, cert_f, label, name);
                /* fprintf(fp,"rule (%d, %s, %s, %d) :- fact(%s, %s).\n",
                      rule_num++, type, ename, (cert_f - 50),
                      label, name2); */       /* use for a 0 to 100 scale */
            fprintf(fp,"rule (%d, %s, %s, -%d) :- fact(%s, %s).\n", rule_num++,
              type, ename, cert_f, label, name2); /* -100:100 scale */
           }        /* end if then */
           else if ((ccell->value == 2) || (ccell->value == 1)) {
            fprintf(fp,"rule (%d, %s, %s, -%d) :- fact(%s, %s).\n", rule_num++,
              type, ename, cert_f, label, name);
                /* fprintf(fp,"rule (%d, %s, %s, %d) :- fact(%s, %s).\n",
                      rule_num++, type, ename, (cert_f - 50),
                      label, name2); */       /* use for a 0 to 100 scale */
            fprintf(fp,"rule (%d, %s, %s, %d) :- fact(%s, %s).\n", rule_num++,
              type, ename, cert_f, label, name2); /* -100:100 scale */
           }        /* end if else */
         }         /* end if (cert_f != 0) */
        celem = celem->next;                   /* move to next element */
        ccell = ccell->next;                   /*  and associated grid value */
      }          /* end while */
    ctrat = ctrat->next;          /* move on to next trat */
  }          /* end while */

  fclose (fp);
  printf("The rule list has been saved in the file %s.\n\n\n\n", filename);
}                  /* end gen_t_prolog (void) */


/*****************************************************>void gen_hc_prolog (void)
* FUNCT : generat a rule base from the grid for hc prolog
* VARS  : NONE
* NOTES : ??
*  1) - This procedure does NOT print out negative numbers, it only
*       inserts a "-" sign before its positive value
*/
void gen_hc_prolog()
{
TITLE *celem, *ctrat;
CELL *ccell;
FILE *fp;
char line[20], filename[14], rulename[14], ans;
char name[16], name2[16], label[16], ename[16];
char desc[16], edesc[16], tdesc[16], iname[16];
long cert_f, trat_f, grid_f, scale;
int pntr, period, index, rule_num;

  printf("Please enter the file name for the rule base.\n");
  strcpy (filename, trat->filename);       /* build rule-base filename */
period = strcspn(filename,".\0");          /* find period */
if (filename[period] == '.') {             /* if present, */
    filename[period] = NULL;                /* delete it and extension */
    strcat (filename, ".HC");               /* add ".HC" extension */
  }              /* end IF */
ans = get_filename (filename, ".HC");
strcpy (rulename, filename);               /* copy predicate name for rules */
period = strcspn(rulename,".\0");          /* find period, if present, delete */
strlwr(rulename);                          /* it and make it all lower case */
if (rulename[period] == '.')  rulename[period] = NULL; /* it and extension */
if (ans == 'F') return;
fp = fopen(filename,"r");      /* if file does not exist, fp = NULL */
if (fp != NULL) {   /* if a file already exists, give the user a warning */
    fclose(fp);                            /* first close file */
    line[0] = NULL;                        /* erase any old info in line */
    printf("The file %s already exists.\n    -=NOTE=- continuing will ",
        filename);
    printf("destroy anything currently in that file.\n             Do you ");
    printf("still want to use this file? (YES or *NO) -> ");
    gets(line);
    printf("\n\n");
    strupr(line);
    if (strstr(line,"YES") == NULL) /* if don't want to use the file */
        filename[0] = NULL;        /*  go back and get another filename */
  }              /* end IF */

line[0] = NULL;                /* erase any old info in line */
printf("Rule generation for HC Prolog is about to begin.  Note that since ");
printf("there are\n    %d %ss and %d %ss, over %d rules could be generated.",
    elem->number, elem->type, trat->number, trat->type,
    elem->number*trat->number*2);
printf("\nWould you like to proceed?   (*YES or NO) -> ");
gets(line);
printf("\n\n");
strupr(line);
if (strstr(line,"NO") != NULL)             /* if don't want rules generated */
    return;

printf("What is a short description of the area you are\n");
printf("investigating?  (enter it here) -> ");
gets(line);                                    /* Chirality */
printf("\n\n");
get_name (desc, line);
printf("In looking at %s, what is a description of the final\n", desc);
printf("    answer you are looking for?  (enter it here) -> ");
gets(line);                                    /* Chiral Column */
printf("\n\n");
get_name (edesc, line);
printf("In classifying the different %ss, what is a description\n", edesc);
printf("    of the traits you are using?   (enter it here) -> ");
gets(line);                                    /* Functional Groups */
printf("\n\n");
get_name (tdesc, line);
printf("If all of the %ss deal with one particular item, what\n", tdesc);
printf("    is a description of that item? (Return for none) -> ");
gets(line);                                    /* Molecule */
printf("\n\n");
get_name (iname, line);
if (line[0] == NULL)               /* if nothing typed,set item name as */
    iname[0] = NULL;               /*  null as well (not done by get_name() */

fp = fopen(filename,"w");
fprintf(fp,";                      ******* File %s *******\n\n", filename);
fprintf(fp,"(assert\n");
fprintf(fp,"; generated using PCT\n\n");
ctrat = trat->head->next;                  /* print fact rules for HC Prolog */
fprintf(fp,"((quoa lst \"%s\" \"%s\" \"%s\" \"%s\" \"%s\" (\n", rulename,
    desc, edesc, tdesc, iname);
while (ctrat != trat->head) {                          /* for all traits */
```

```
      strcpy (label,ctrat->label); ch2_hc_prolog(label); /* change output to */
      strcpy (name,ctrat->name); ch2_hc_prolog(name);    /* all lower case as*/
      strcpy (name2,ctrat->name2); ch2_hc_prolog(name2); /* t-prolog requires*/
      fprintf(fp,"     (%s %s %s)\n", label, name, name2);
      ctrat = ctrat->next;          /* move on to next trat */
    }          /* end while */
  fprintf(fp,"                 ))\n\n");
  fprintf(fp,"((maxrules %d))\n\n", elem->number*trat->number*2);

  ctrat = trat->head->next;          /* print knowledge rules for HC Prolog */
  rule_num = 1;
  scale = scale_factor (trat->number);        /* get scaling factor */
  while (ctrat != trat->head) {               /* for all traits */
      trat_f = (ctrat->weight * 2);           /* calculate trait factor */
      celem = elem->head->next;
      ccell = ctrat->grid;
      while (celem != elem->head) {           /* for all elements */
          if ((ccell->value == 5) || (ccell->value == 1))
              grid_f = 10;                    /* assign grid factor */
            else if ((ccell->value == 4) || (ccell->value == 2))
                    grid_f = 5;
                  else grid_f = 0;
/*          cert_f = (((trat_f * grid_f * scale) + 50) / 100); */
        cert_f = (((trat_f * grid_f * scale) + 500) / 1000);
                /* calculate certainty factor, range from -100:100 */
       /* cert_f = (cert_f / 2) + 50;     map range from -100:100 to 0:100 */
        if (cert_f != 0) {                  /* output rules - change output */
          strcpy (ename,celem->name);       /* to all lower case as hc */
          ch2_hc_prolog (ename);            /*  prolog requires */
          strcpy (label,ctrat->label);  ch2_hc_prolog (label);
          strcpy (name,ctrat->name);    ch2_hc_prolog (name);
          strcpy (name2,ctrat->name2);  ch2_hc_prolog (name2);
          if ((ccell->value == 5) || (ccell->value == 4)) {
            fprintf(fp,"((%s %d %s %ld) if (fact %s %s))\n", rulename,
              rule_num++, ename, cert_f, label, name);
                /* fprintf(fp,"rule (%d, %s, %d) :- fact(%s, %s).\n",
                      rule_num++, ename, (cert_f - 50),
                      label, name2); */       /* use for a 0 to 100 scale */
            fprintf(fp,"((%s %d %s -%ld) if (fact %s %s))\n", rulename,
              rule_num++, ename, cert_f, label, name2);
           }         /* end if then */
           else if ((ccell->value == 2) || (ccell->value == 1)) {
            fprintf(fp,"((%s %d %s -%ld) if (fact %s %s))\n", rulename,
              rule_num++, ename, cert_f, label, name);
                /* fprintf(fp,"rule (%d, %s, %d) :- fact(%s, %s).\n",
                      rule_num++, ename, (cert_f - 50),
                      label, name2); */       /* use for a 0 to 100 scale */
            fprintf(fp,"((%s %d %s %ld) if (fact %s %s))\n", rulename,
              rule_num++, ename, cert_f, label, name2);
          }          /* end if else */
        }          /* end if (cert_f != 0) */
        celem = celem->next;                /* move to next element */
        ccell = ccell->next;                /*  and associated grid value */
      }          /* end while */
      ctrat = ctrat->next;          /* move on to next trat */
    }          /* end while */

  fprintf(fp,"\n)\n;  *** END OF assert\n");
  fclose (fp);
  printf("The rule list has been saved in the file %s.\n\n\n\n", filename);
}                 /* end gen_hc_prolog() */


/*****************************************************>void gen_prolog (void)
* FUNCT : generat a rule base w/entailments from the grid for prolog
* VARS  : NONE
* NOTES : ??
*  1) - This procedure does NOT print out negative numbers, it only
*       inserts a " " sign before its positive value
*/
void gen_prolog()
{
TITLE *celem, *ctrat, *trata, *tratb;
CELL *ccell, *cella, *cellb;
FILE *fp;
char line[20], filename[14], ans;
int AB, AB_, A_B, A_B_, ABN, AB_N, A_BN, A_B_N;
int pntr, period, index, rule_num, trat_f, grid_f, cert_f;

  printf("Please enter the file name for the rule base.\n");
  strcpy (filename, trat->filename);       /* build rule-base filename */
  period = strcspn(filename,".\0");        /* find period */
  if (filename[period] == '.') {           /* if present, */
      filename[period] = NULL;                 /* delete it and extension */
      strcat (filename, ".RUL");               /* add ".RUL" extension */
    }                /* end IF */
  ans = get_filename (filename, ".RUL");
  if (ans == 'F') return;
  fp = fopen(filename,"r");      /* if file does not exist, fp = NULL */
  if (fp != NULL) {    /* if a file already exists, give the user a warning */
      fclose(fp);                          /* first close file */
      printf("The file %s already exists.\n    -=NOTE=- continuing will ",
          filename);
      printf("destroy anything currently in that file.\n             Do you ");
      printf("still want to use this file? (YES or *NO) -> ");
      gets(line);
      printf("\n\n");
      strupr(line);
      if (strstr(line,"YES") == NULL) /* if don't want to use the file */
          filename[0] = NULL;        /*  go back and get another filename */
    }                /* end IF */

  printf("Rule generation for Prolog (including entailments) is about to begin");
  printf(".  Note that\n    since there are %d %ss and %d %ss, over %d rules ",
      elem->number, elem->type, trat->number, trat->type,
      elem->number*trat->number*2);
  printf("could be Generated.\nWould you like to proceed?   (*YES or NO) -> ");
  gets(line);
  printf("\n\n");
  strupr(line);
  if (strstr(line,"NO") != NULL)           /* if don't want rules generated */
      return;

  fp = fopen(filename,"w");
  ctrat = trat->head->next;                 /* print fact rules for Prolog */
  fprintf(fp,"((QUES_LST (\n");
  while (ctrat != trat->head) {                        /* for all traits */
      fprintf(fp,"     (%s %s %s)\n", ctrat->label, ctrat->name, ctrat->name2);
      ctrat = ctrat->next;          /* move on to next trat */
    }          /* end while */
  fprintf(fp,"                 )))\n\n");
  fprintf(fp,"( (MAXRULES %d) )\n\n", elem->number*trat->number*2);
  fprintf(fp,"\n");

  trata = trat->head->next;                 /* print entailment rules */
  while (trata != trat->head) /* while 'A' */ {      /* Do for all traits 'a' */
    tratb = trat->head->next;
    while (tratb != trat->head ) /* while 'B' */ {  /* Do for all traits 'b' */
      if (trata == tratb )                          /* if 'a' = 'b', go on to */
        { tratb = tratb->next; continue; }          /*   next trat and repeat */
      cella = trata->grid;                           /* compare all cells */
      cellb = tratb->grid;
      AB = 0; AB_ = 0; A_B = 0; A_B_ = 0;            /* initialize variables */
      ABN = 0; AB_N = 0; A_BN = 0; A_B_N = 0;
      do {
          switch (cella->value) {
            case 5 : switch (cellb->value) {
                       case 5 : AB = AB + 100; ABN++; break;
```

```
                    case 4 : AB = AB + 75; ABN++; break;
                    case 2 : AB_ = AB_ + 75; AB_N++; break;
                    case 1 : AB_ = AB_ + 100; AB_N++;
                   } break;
          case 4 : switch (cellb->value) {
                    case 5 :
                    case 4 : AB = AB + 100; ABN++; break;
                    case 2 :
                    case 1 : AB_ = AB_ + 100; AB_N++;
                   } break;
          case 2 : switch (cellb->value) {
                    case 5 :
                    case 4 : A_B = A_B + 100; A_BN++; break;
                    case 2 :
                    case 1 : A_B_ = A_B_ + 100; A_B_N++;
                   } break;
          case 1 : switch (cellb->value) {
                    case 5 : A_B = A_B + 100; A_BN++; break;
                    case 4 : A_B = A_B + 75; A_BN++; break;
                    case 2 : A_B_ = A_B_ + 75; A_B_N++; break;
                    case 1 : A_B_ = A_B_ + 100; A_B_N++;
                   } break;
          }     /* end switch */
        cella = cella->next;
        cellb = cellb->next;
      } while (cella != trata->grid);     /* end do while */
    if (AB != 0) {
      AB = ((((AB *10) / ABN) + 5) / 10);  /* round off result */
      fprintf(fp,"( (%s %s %d) if (%s %s))\n", tratb->label, tratb->name,
            AB, trata->label, trata->name);
    }       /* end if */
    if (AB_ != 0) {
      AB_ = ((((AB_ *10) / AB_N) + 5) / 10);  /* round off result */
      fprintf(fp,"( (%s %s %d) if (%s %s))\n", tratb->label, tratb->name2,
            AB_, trata->label, trata->name);
    }       /* end if */
    if (A_B != 0) {
      A_B = ((((A_B *10) / A_BN) + 5) / 10);  /* round off result */
      fprintf(fp,"( (%s %s %d) if (%s %s))\n", tratb->label, tratb->name,
            A_B, trata->label, trata->name2);
    }       /* end if */
    if (A_B_ != 0) {
      A_B_ = ((((A_B_ *10) / A_B_N) + 5) / 10);  /* round off result */
      fprintf(fp,"( (%s %s %d) if (%s %s))\n", tratb->label, tratb->name2,
            A_B_, trata->label, trata->name2);
    }       /* end if */
    tratb = tratb->next;
  }  /* end while 'B' */
 trata = trata->next;
}  /* end while 'A' */
fprintf(fp,"\n");

ctrat = trat->head->next;                       /* print knowledge rules */
rule_num = 1;
while (ctrat != trat->head) {                   /* for all traits */
    trat_f = (ctrat->weight * 2) * 10;          /* calculate trait factor */
    celem = elem->head->next;
    ccell = ctrat->grid;
    while (celem != elem->head) {               /* for all elements */
        if ((ccell->value == 5) || (ccell->value == 1))
            grid_f = 100;                       /* assign grid factor */
          else if ((ccell->value == 4) || (ccell->value == 2))
                 grid_f = 50;
               else grid_f = 0;
        cert_f = (trat_f * grid_f) / 100;     /* calculate certainty factor */
        cert_f = (cert_f / 2) + 50;     /* map scale from -100:100 to 0:100 */
        if (cert_f != 0) {                      /* output rules */
            fprintf(fp,"( (RULE %d (%s %s %d)) if (%s %s))\n", rule_num++,
                elem->type, celem->name, cert_f, ctrat->label, ctrat->name);
            fprintf(fp,"( (RULE %d (%s %s %d)) if (%s %s))\n", rule_num++,
                elem->type, celem->name, (100 - cert_f), ctrat->label,
                ctrat->name2);
        }       /* end if */
      celem = celem->next;                      /* move to next element */
      ccell = ccell->next;                      /*  and associated grid value */
    }       /* end while */
  ctrat = ctrat->next;          /* move on to next trat */
 }      /* end while */

 fclose (fp);
 printf("The rule list has been saved in the file %s.\n\n\n\n", filename);
}               /* end gen_prolog() */


/*********************************************************>int auto_anal (void)
* FUNCT : perform the analysis and generation of the rule base from the grid
* VARS  : NONE
* RETNS : returning '0' indicates successful completion, any other
*         value indicates some kind of error.
* NOTES : ??
*/
int auto_anal()
{
DOUBLE *dbl;
char line[20], cmd_ary[MAX_CMDS][10];
int num_cmds, cmd_num;

  printf("     The procedure for automated analysis is about to begin.  This ");
  printf("procedure\nwill take you through the analysis of both the element ");
  printf("and traits to resolve\nmany inconsistencies found to be present.  ");
  printf("Following this procedure, options\nwill be presented to allow ");
  printf("generation of a rule-based system reflecting\nthe information ");
  printf("contained in the grid.\n\n\nPlease hit RETURN to continue - ");
  gets(line);
  printf("\n\n");
  ecluster_grid();
  dbl = m_list->clstr_e; /* note - highest match is first in list */
  make_cmd_list (*COMBINE INPUT VERIFY GO-ON FINISHED RPT-EMAT LIST-G QUIT HELP",
      &num_cmds, cmd_ary);
  while (dbl != NULL) {
    printf("     It has been determined that the two elements {%s} and\n",
      find_name (elem,dbl->item_a));
    printf("{%s} are %d percent alike.  In order to resolve this similarity,\n",
      find_name(elem,dbl->item_b), dbl->score);
    printf("you can COMBINE them into a new element, or INPUT another trait ");
    printf("to\nfurther distinguish between them.\n\n");
    help_cmd_list(elem->type,num_cmds, cmd_ary);
    printf("Please enter your choice here -> ");
    gets(line);
    printf("\n\n\n");
    cmd_num = check4cmd (line, cmd_ary, num_cmds);
    switch (cmd_num) {
          /* COMBINE INPUT VERIFY GO-ON FINISHED RPT-EMAT LIST-G QUIT HELP */
        case 0 :        /* no commands match with users input */
                printf("The command you entered, %s, is not understood.",
                    line);
                printf("  Please enter\n            your choice",
                    " again.\n\n\n\n");
                break;
        case 1 : combine_elem (dbl);
                break;
        case 2 : printf("Please input a new %s that will distinguish between ",
                    trat->type);
                printf("the two\n     %ss {%s} and {%s}.\n\n",elem->type,
                    find_name(elem,dbl->item_a), find_name(elem,dbl->item_b));
                add_node(trat, trat->head, (trat->number + 1));
                break;                          /* add to end of trat list */
        case 3 : verify_elem (dbl);
```

```
            break;
      case 4 : dbl = dbl->next; /* go on to next reported match */
            break;
      case 5 : if (dbl != NULL) {
                 printf("There are still inconsistencies detected amongst");
                 printf(" the elements.  Are\n     you sure you are");
                 printf(" finished? (YES or NO*) -> ");
                 gets(line);
                 strupr(line);
                 printf("\n\n\n");
                 if (strstr(line,"YES") != NULL)
                     dbl = NULL;
               }     /* end if */
            break;
      case 6 : if (m_list->dbl_e != NULL) {
                 printf("NOTE - Correlation");
                 report_ematch (m_list->dbl_e); }
               if (m_list->clstr_e != NULL) {
                 printf("NOTE - Cluster");
                 report_ematch (m_list->clstr_e); }
            break;
      case 7 : list_grid();
            break;
      case 8 : printf("\n\n\n\n\n"); return(1);
            break;
      case 9 : break;/* HELP - the command list will print automatically */
     default : printf("ERROR - analysis command not accounted for!!!");
            gets(line);
            printf("\n\n\n\n\n");
    }       /* end switch */
  if ((cmd_num == 1) || (cmd_num == 2) || (cmd_num == 3)) { /* if grid was */
    del_match_list (&m_list->clstr_e); /* changed, delete current match */
    eclustergrid();                    /* list and check for matches again */
    dbl = m_list->clstr_e;         /* note - highest match is first in list */
   }      /* end if */
 }       /* end while */
tclustergrid();
dbl = m_list->clstr_t; /* note - highest match is first in list */
make_cmd_list ("COMBINE VERIFY GO-ON FINISHED RPT-TMAT LIST-G QUIT HELP",
    &num_cmds, cmd_ary);
while (dbl != NULL) {
  printf("     It has been determined that the two traits {%s} and\n",
    find_name (trat,dbl->item_a));
  printf("{%s} are %d percent alike.  In order to resolve this similarity,\n",
    find_name(trat,dbl->item_b), dbl->score);
  printf("you can COMBINE them into a new trait, or VERIFY the values ");
  printf("entered for each,\nchanging any that are incorrect.\n\n");
  help_cmd_list(trat->type,num_cmds, cmd_ary);
  printf("Please enter your choice here -> ");
  gets(line);
  printf("\n\n\n");
  cmd_num = check4cmd (line, cmd_ary, num_cmds);
  switch (cmd_num) {/*COMBINE VERIFY GO-ON FINISHED RPT-TMAT LIST-G QUIT HELP*/
      case 0 :        /* no commands match with users input */
               printf("The command you entered, %s, is not understood.",
                   line);
               printf("  Please enter\n          your choice again.\n\n\n\n\n");
               break;
      case 1 : printf(" COMBINING\n\n");
               break;
      case 2 : verify_trat (dbl);
               break;
      case 3 : dbl = dbl->next; /* go on to next reported match */
               break;
      case 4 : if (dbl != NULL) {
                 printf("There are still inconsistencies detected amongst");
                 printf(" the elements.  Are\n     you sure you are");
                 printf(" finished? (YES or NO*) -> ");
                 gets(line);
                 strupr(line);
                 printf("\n\n\n");
                   if (strstr(line,"YES") != NULL)
                     dbl = NULL;
               }     /* end if */
            break;
      case 5 : if (m_list->dbl_t != NULL) {
                 printf("NOTE - Correlation");
                 report_tmatch (m_list->dbl_t); }
               if (m_list->clstr_t != NULL) {
                 printf("NOTE - Cluster");
                 report_tmatch (m_list->clstr_t); }
            break;
      case 6 : list_grid();
            break;
      case 7 : printf("\n\n\n\n\n"); return(1);
            break;
      case 8 : break;/* HELP - the command list will print automatically */
     default : printf("ERROR - analysis command not accounted for!!!");
            gets(line);
            printf("\n\n\n\n\n");
    }       /* end switch */
  if ((cmd_num == 1) || (cmd_num == 2)) { /* if a change to grid was made */
    del_match_list (&m_list->clstr_t);      /* delete current match list */
    tclustergrid();                         /* and check for matches again */
    dbl = m_list->clstr_t;         /* note - highest match is first in list */
   }      /* end if */
 }       /* end while */
rate_traits();
rule_gen();
}               /* end auto_anal() */


/*************************************************>char *find_name (item, num)
* FUNCT : To find the name of the num'th item on the item list.
* VARS  : item -> pointer to the item list
*       : num -> the position of the item of interest in the item list
* RETNS : Return a pointer to the num'th item's name in the item list
* NOTES : ??
*/
char *find_name (item, num)
ITEM *item;
int num;
{
TITLE *node;
int index;
  node = item->head->next;
  for (index = 1;index < abs(num);index++)
      node = node->next;
  return(node->name);
}               /* end *find_name (item,num) */


/*************************************************>char *find_name2 (item, num)
* FUNCT : To find the name2 of the num'th item on the item list.
* VARS  : item -> pointer to the item list
*       : num -> the position of the item of interest in the item list
* RETNS : Return a pointer to the num'th item's name2 in the item list
* NOTES : ??
*/
char *find_name2 (item, num)
ITEM *item;
int num;
{
TITLE *node;
int index;
  node = item->head->next;
  for (index = 1;index < abs(num);index++)
      node = node->next;
```

```
  return(node->name2);
}                   /* end *find_name2 (item,num) */


/*************************************************>char *find_label (item, num)
* FUNCT : To find the label of the num'th item on the item list.
* VARS  : item -> pointer to the item list
*       : num -> the position of the item of interest in the item list
* RETNS : Return a pointer to the num'th item's label in the item list
* NOTES : ??
*/
char *find_label (item, num)
ITEM *item;
int num;
{
TITLE *node;
int index;
  node = item->head->next;
  for (index = 1;index < abs(num);index++)
      node = node->next;
  return(node->label);
}                   /* end *find_label (item,num) */


/*************************************************>void combine_elem (dbl)
* FUNCT : combine the two matching elements noted in the match list
* VARS  : dbl -> pointer to the match list
* NOTES : ??
*/
void combine_elem (dbl)
DOUBLE *dbl;
{
TITLE *new_node, *ctrat, *elema, *elemb;
CELL *cella, *cellb, *newcell;
char line[30];
int index, item_num, number;

  printf("The two elements that are %d percent alike are listed below :\n",
      dbl->score);
  elema = elem->head->next;           /* find the first matching element */
  for (index = 1;index < dbl->item_a;index++) elema = elema->next;
  elemb = elem->head->next;           /* find the second matching element */
  for (index = 1;index < dbl->item_b;index++) elemb = elemb->next;
  printf("            NAME :   %-15s     %-15s\n", elema->name, elemb->name);
  printf("     ABBREVIATION:   %-15s     %-15s\n", elema->name2, elemb->name2);
  printf("\nThese two elements are to be combine? (YES* or NO) ");
  gets(line);
  strupr(line);
  if (strstr(line,"NO") != NULL) return;

  new_node = (TITLE *)malloc(sizeof(TITLE));
  new_node->last = elem->head->last;/* attach 'new_node' at end of elem list */
  new_node->next = elem->head;
  elem->head->last->next = new_node;
  elem->head->last = new_node;
  ++elem->number;          /* elem_num now holds the position of the new node */
  new_node->grid = NULL;
  new_node->weight = 0;
  new_node->name[0] = NULL;        /* initialize character fields */
  new_node->name2[0] = NULL;
  new_node->label[0] = NULL;
  add_elem_cells (elem->number);

  printf("\n\nWhat is the name for the new element? ");
  gets(line);
  get_name (new_node->name, line);
  printf("\n     What is the abbreviation for {%s}? ", new_node->name);
  gets(line);
  get_name (new_node->name2, line);
  new_node->name2[3] = NULL;
  printf("\n\nNow rate {%s} on each of the following %s.  The current values",
      new_node->name, trat->type);
  printf("\n     of the two %ss that are being combine are shown on the\n",
      elem->type);
  printf("     left-hand side of each line. \n\n");
  printf(" %3s = %s\n", elema->name2, elema->name);
  printf("    %3s = %-15s       Current %-15s       %15s\n",
      elemb->name2, elemb->name, trat->type, new_node->name);
  for (index = 1;index < 79;index++) printf("="); printf("\n");
  ctrat = trat->head->next;
  while (ctrat != trat->head) {
      cella = ctrat->grid;
      for (index = 1;index < dbl->item_a;index++)
          cella = cella->next;
      cellb = ctrat->grid;
      for (index = 1;index < dbl->item_b;index++)
          cellb = cellb->next;
      newcell = ctrat->grid->last;
      printf("  %d  %d - Value for %s [%s=5, %s=1] -> ", cella->value,
          cellb->value, ctrat->label, ctrat->name, ctrat->name2);
      scanf("%d",&number);
      gets(line);  /* clear newline char */
      if ((number < 1) || (number > 5)) {
          printf("    Please enter a number between 1 and 5.\n");
          continue;
        }          /* end if */
      newcell->value = number;
      ctrat = ctrat->next;
    }             /* end while */
  printf("\n\n");
  remove_node (elem, elema, dbl->item_a);
  remove_node (elem, elemb, dbl->item_b);
}          /* end combine_elem (dbl) */


/*************************************************>void verify_elem (dbl)
* FUNCT : verify the values entered for the two matching elements
* VARS  : dbl -> pointer to the match list
* NOTES : ??
*/
void verify_elem (dbl)
DOUBLE *dbl;
{
TITLE *elema, *elemb;
int index;

  elema = elem->head->next;
  for (index =1;index < dbl->item_a;index++)
      elema = elema->next;
  printf("Please verify the current data for the %s {%s}:\n\n", elem->type,
      elema->name);
  ch_elem_cells (elema, dbl->item_a);
  elemb = elem->head->next;
  for (index =1;index < dbl->item_b;index++)
      elemb = elemb->next;
  printf("\n\n\nPlease verify the current data for the %s {%s}:\n\n",
      elem->type, elemb->name);
  ch_elem_cells (elemb, dbl->item_b);
  printf("\n\n\n");

}          /* end verify_elem (dbl) */


/*************************************************>void verify_trat (dbl)
* FUNCT : verify the values entered for the two matching traits
* VARS  : dbl -> pointer to the match list
* NOTES : ??
*/
```

```
void verify_trat (dbl)
DOUBLE *dbl;
{
TITLE *trata, *tratb;
int index;

  trata = trat->head->next;
  for (index =1;index < dbl->item_a;index++)
      trata = trata->next;
  printf("Please verify the current data for the %s {%s}:\n\n", trat->type,
      trata->name);
  ch_trat_cells (trata);
  tratb = trat->head->next;
  for (index =1;index < dbl->item_b;index++)
      tratb = tratb->next;
  printf("Please verify the current data for the %s {%s}:\n\n", trat->type,
      tratb->name);
  ch_trat_cells (tratb);
  printf("\n\n\n");

}         /* end verify_trat (dbl) */
```

APPENDIX D: KEY Output File Formats

Appendix D contains examples of the four files generated by KEY: the element (.ELM), trait (.TRT), grid (.GRD), and rulebase (.HC) files. The file formats, as well as their corresponding data structures, are described in Section IV.F. The examples given are taken form the OPTIMER system described in Section IV.B.

```
9 0
element abbreviation .ELM
P4                P4
P5-7              P5
P8-9              P8
P10-11            P10
P12               P12
P13               P13
P14               P14
P15               P15
P16               P16


6 0
trait opposite .TRT
Benz              No-Benz          Aromatic
CdO               No-CdO           Carbonyl
OsH               No-OH            Alchol
N3                No-N3            Tert-Nitrogen
NP                No-NP            Primary-Amine
COOH              No-CO2H          Carboxylic-Acid


6 9
trait opposite .GRD
Benz              No-Benz          Aromatic
 5 5 5 5 5 5 1 1 1
CdO               No-CdO           Carbonyl
 5 5 5 1 1 1 3 3 3
OsH               No-OH            Alchol
 5 1 1 5 1 1 3 3 3
N3                No-N3            Tert-Nitrogen
 3 5 1 3 5 1 5 3 3
NP                No-NP            Primary-Amine
 3 3 3 3 3 3 5 1 1
COOH              No-CO2H          Carboxylic-Acid
 3 3 3 3 3 3 3 5 1


;                    ******* File CHIRALIB.HC *******

(assert
; generated using PCT

((ques_lst "chiralib" "Chirality" "Chiral-Column" "Compound-Trait" "Molecule" (
     (Aromatic Benz No-Benz)
     (Carbonyl CdO No-CdO)
     (Alchol OsH No-OH)
     (Tert-Nitrogen N3 No-N3)
     (Primary-Amine NP No-NP)
     (Carboxylic-Acid COOH No-CO2H)
          )     ))

((maxrules 108))

((chiralib 1 P4 44) if (fact Aromatic Benz))
((chiralib 2 P4 -44) if (fact Aromatic No-Benz))
((chiralib 3 P5-7 44) if (fact Aromatic Benz))
((chiralib 4 P5-7 -44) if (fact Aromatic No-Benz))
((chiralib 5 P8-9 44) if (fact Aromatic Benz))
((chiralib 6 P8-9 -44) if (fact Aromatic No-Benz))
((chiralib 7 P10-11 44) if (fact Aromatic Benz))
((chiralib 8 P10-11 -44) if (fact Aromatic No-Benz))
((chiralib 9 P12 44) if (fact Aromatic Benz))
((chiralib 10 P12 -44) if (fact Aromatic No-Benz))
((chiralib 11 P13 44) if (fact Aromatic Benz))
((chiralib 12 P13 -44) if (fact Aromatic No-Benz))
((chiralib 13 P14 -44) if (fact Aromatic Benz))
((chiralib 14 P14 44) if (fact Aromatic No-Benz))
((chiralib 15 P15 -44) if (fact Aromatic Benz))
((chiralib 16 P15 44) if (fact Aromatic No-Benz))
((chiralib 17 P16 -44) if (fact Aromatic Benz))
((chiralib 18 P16 44) if (fact Aromatic No-Benz))
((chiralib 19 P4 44) if (fact Carbonyl CdO))
((chiralib 20 P4 -44) if (fact Carbonyl No CdO))
((chiralib 21 P5-7 44) if (fact Carbonyl CdO))
((chiralib 22 P5-7 -44) if (fact Carbonyl No-CdO))
((chiralib 23 P8-9 44) if (fact Carbonyl CdO))
((chiralib 24 P8-9 -44) if (fact Carbonyl No-CdO))
((chiralib 25 P10-11 -44) if (fact Carbonyl CdO))
((chiralib 26 P10-11 44) if (fact Carbonyl No-CdO))
((chiralib 27 P12 -44) if (fact Carbonyl CdO))
((chiralib 28 P12 44) if (fact Carbonyl No-CdO))
((chiralib 29 P13 -44) if (fact Carbonyl CdO))
((chiralib 30 P13 44) if (fact Carbonyl No-CdO))
((chiralib 31 P4 44) if (fact Alchol OsH))
((chiralib 32 P4 -44) if (fact Alchol No-OH))
((chiralib 33 P5-7 -44) if (fact Alchol OsH))
((chiralib 34 P5-7 44) if (fact Alchol No-OH))
((chiralib 35 P8-9 -44) if (fact Alchol OsH))
((chiralib 36 P8-9 44) if (fact Alchol No-OH))
((chiralib 37 P10-11 44) if (fact Alchol OsH))
((chiralib 38 P10-11 -44) if (fact Alchol No-OH))
((chiralib 39 P12 -44) if (fact Alchol OsH))
((chiralib 40 P12 44) if (fact Alchol No-OH))
((chiralib 41 P13 -44) if (fact Alchol OsH))
((chiralib 42 P13 44) if (fact Alchol No-OH))
((chiralib 43 P5-7 44) if (fact Tert-Nitrogen N3))
((chiralib 44 P5-7 -44) if (fact Tert-Nitrogen No-N3))
((chiralib 45 P8-9 -44) if (fact Tert-Nitrogen N3))
((chiralib 46 P8-9 44) if (fact Tert-Nitrogen No-N3))
((chiralib 47 P12 44) if (fact Tert-Nitrogen N3))
((chiralib 48 P12 -44) if (fact Tert-Nitrogen No-N3))
((chiralib 49 P13 -44) if (fact Tert-Nitrogen N3))
((chiralib 50 P13 44) if (fact Tert-Nitrogen No-N3))
((chiralib 51 P14 44) if (fact Tert-Nitrogen N3))
((chiralib 52 P14 -44) if (fact Tert-Nitrogen No-N3))
((chiralib 53 P14 44) if (fact Primary-Amine NP))
((chiralib 54 P14 -44) if (fact Primary-Amine No-NP))
((chiralib 55 P15 -44) if (fact Primary-Amine NP))
((chiralib 56 P15 44) if (fact Primary-Amine No-NP))
((chiralib 57 P16 -44) if (fact Primary-Amine NP))
((chiralib 58 P16 44) if (fact Primary-Amine No-NP))
((chiralib 59 P15 44) if (fact Carboxylic-Acid COOH))
((chiralib 60 P15 -44) if (fact Carboxylic-Acid No-CO2H))
((chiralib 61 P16 -44) if (fact Carboxylic-Acid COOH))
((chiralib 62 P16 44) if (fact Carboxylic-Acid No-CO2H))

)
;  *** END OF assert
```

APPENDIX E: KEY Inference Engine Code

Appendix E lists the source code for the KEY inference engine. There are two different versions of the inference engine: one for VPI Prolog, and the other for Borland's TurboProlog. Section IV.E explains how the inference engine reads the rules and calculates and orders the results. Section IV.B and Appendix A demonstrate how to run the inference engine. Refer also to Appendix F, since KEY is usually used in conjunction with CHESS.

```
;                       **** File KEY.HC ****                 February 26, 1991
dynamic cf
dynamic fact

(assert

; (akey) - abolish all predicates in KEY.HC so you can try again
          ((akey) if
                        (abolish run)                                   (abolish run_rb)
                        (abolish test_rb)
                        (abolish test1_rb)              (abolish test2_rb)
                        (abolish test3_rb)              (abolish test4_rb)
                        (abolish ld_rulebase)  (abolish ask4item)
                        (abolish check4mol)
                                ;*** This must be changed for each Rulebase ***
                        (abolish ask_user)              (abolish answer)
                        (abolish split)                 (abolish split2)
                        (abolish split3)                (abolish exrule)
                        (abolish search_rules) (abolish testrule)
                        (abolish compute_cf)   (abolish printresults)
                        (abolish findlargest)  (abolish deletefacts)
                        (abolish list-cat)
                        (abolish akey)                                  (consult key.hc)
          )

;  **Format of Rules in PCT rulebase
;          (ques_let ?rulebase ?descr ?elemdesc ?tratdesc ?itemdesc ?Namelist)
;          (maxrules ?rnum)
;          ((?rulebase ?rnum ?cname) if (fact ?desc tname))
;  **Format of Rules in dynamic rulebase
;          (fact ?tdesc ?tname)              ;Stores answers for Users Questions
;          (cf ?Ele ?Cf)                     ;Stores CF's for each element

; (run) - load the PCT rulebase and call predicate "run_rb" to run it.
; NOTE - Predicate "ask4item" is taylored for use with the hooks made
;          available by CHESS.  Refer to NOTE for predicate "ask4item".
  ((run) if
          (println "What is the name of the rule base you want to use?")
          (print "      Please enter it here -> ")
          (getline stdin ?rb)
          (println)
          (:= ?rulebase [strcase ?rb lower])
          (ld_rulebase ?rulebase)
          (ques_let ?rulebase ?descr ?elemdesc ? ?itemdesc ?)
          (ask4item ?descr ?elemdesc ?itemdesc ?iname)  ;get item name if present
          (run_rb ?rulebase ?iname)
  )

; (run_rb ?rulebase ?iname) - Run the PCT rulebase ?rulebase with the item
;          name ?iname if it is present.
  ((run_rb ?rulebase ?iname) if
          (ques_let ?rulebase ?descr ?elemdesc ?tratdesc ?itemdesc ?Namelist)
          (print "We are currently looking at " ?descr ", trying to decide ")
            (print "on\n      which " ?elemdesc " to choose based on ")
            (print ?tratdesc ".\n       (" ?itemdesc " name is ->")
            (println ?iname "<- )\n")
          (list-cat (?descr) ( : ) ?Name2)
          (ask_user ?Namelist ?Name2 ?iname ?Namef)
          (search_rules ?rulebase 1)
          (cut)
          (print "We are currently looking at " ?descr ", trying to decide ")
            (print "on\n      which " ?elemdesc " to choose based on ")
            (print ?tratdesc ".\n                (" ?itemdesc " name is ->")
            (println ?iname "<- )")
          (printresults)
          (println "The following is the data for " ?Namef ".")
          (deletefacts ?Namelist)
          (cut)
    )

; (test_rb) - Test a number of molecules against the OPTIMER rulebase that
;          is entered below.  Note that predicate "ask4item" is not used since
;          the item names (the molecule names) are already provided.  The molecule
;          names MUST agree with the names in the Molecule KB.  If they don't agree,
;          the predicate will fail.  Since predicate "ask4item" isn't used, no call
;          is made to predicate "check4mol", and hence no testing of the name is
;          done.  The print statement with the correct page number is provided for
;          comparison.
;          (run_rb ?rulebase "")
;           (println "***** is found on Page .")
  ((test_rb) if
          (println "\n\nWhat is the name of the rule base you want to run tests on?")
          (print "      Please enter it here -> chiral-b")
          (println)
          (ld_rulebase "chiral-b")
          (ques_let ?rulebase ?descr ?elemdesc ?tratdesc ?itemdesc ?) (cut)
          (run_rb ?rulebase "MANDELAMIDE") (cut)
           (println "***** MANDELAMIDE is found on Page 4.\n\n")
          (run_rb ?rulebase "PRACTOLOL") (cut)
           (println "***** PRACTOLOL is found on Page 4.\n\n")
          (run_rb ?rulebase "2 BUTYL PHENYL CARBAMATE") (cut)
           (println "***** 2 BUTYL PHENYL CARBAMATE is found on Page 4.\n\n")
          (run_rb ?rulebase "D-PHENYLALANINE") (cut)
           (println "***** D-PHENYLALANINE is found on Page 5.\n\n")
          (run_rb ?rulebase "HISTIDINE") (cut)
           (println "***** HISTIDINE is found on Page 5.\n\n")
          (run_rb ?rulebase "2-AMINO-3-PHENYLPROPIONAMIDE") (cut)
           (println "***** 2-AMINO-3-PHENYLPROPIONAMIDE is found on Page 5.\n\n")
          (run_rb ?rulebase "TRIADIMEFON") (cut)
           (println "***** TRIADIMEFON is found on Page 6.\n\n")
          (run_rb ?rulebase "ACETYLPHENETHOLIDE") (cut)
           (println "***** ACETYLPHENETHOLIDE is found on Page 6.\n\n")
          (run_rb ?rulebase "N-(PHENYLETHYL)ACETAMIDE") (cut)
           (println "***** N-(PHENYLETHYL)ACETAMIDE is found on Page 6.\n\n")
          (run_rb ?rulebase "N,O-DIBENZOYL2-AMINORUTANOL") (cut)
           (println "***** N,O-DIBENZOYL2-AMINOBUTANOL is found on Page 7.\n\n")
          (run_rb ?rulebase "FENOPROFEN") (cut)
           (println "***** FENOPROFEN is found on Page 8.\n\n")
          (run_rb ?rulebase "METHYL-3-BENZOYLOXYPROPANOATE") (cut)
           (println "***** METHYL-3-BENZOYLOXYPROPANOATE is found on Page 8.\n\n")
          (run_rb ?rulebase "FLURIPROFEN ME ESTER") (cut)
           (println "***** FLURIPROFEN ME ESTER is found on Page 8.\n\n")
          (run_rb ?rulebase "METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE") (cut)
           (print "***** METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE is ")
           (println "on Page 9.\n\n") (cut)
          (run_rb ?rulebase "METHYL2-PHENYLPROPIONATE") (cut)
           (println "***** METHYL2-PHENYLPROPIONATE is found on Page 9.\n\n")
          (run_rb ?rulebase "1 METHYL 3 PHENYL PROPYL BENZOATE") (cut)
           (println "***** 1 METHYL 3 PHENYL PROPYL BENZOATE is found on Page 9.\n\n")
          (run_rb ?rulebase "GUALFENESINE") (cut)
           (println "***** GUALFENESINE is found on Page 10.\n\n")
          (run_rb ?rulebase "9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE") (cut)
           (println "***** 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE is ")
           (println "on Page 9.\n\n")
          (run_rb ?rulebase "1-PHENYL-1,2-ETHANEDIOL") (cut)
           (println "***** 1-PHENYL-1,2-ETHANEDIOL is found on Page 10.\n\n")
          (run_rb ?rulebase "PERISOXAL") (cut)
           (println "***** PERISOXAL is found on Page 11.\n\n")
          (run_rb ?rulebase "PINDOLOL") (cut)
           (println "***** PINDOLOL is found on Page 11.\n\n")
          (run_rb ?rulebase "METOPROLOL") (cut)
           (println "***** METOPROLOL is found on Page 11.\n\n")
          (run_rb ?rulebase "CARRINOXAMINE") (cut)
           (println "***** CARRINOXAMINE is found on Page 12.\n\n")
          (run_rb ?rulebase "1-PHENYLETHYLAMINE") (cut)
           (println "***** 1-PHENYLETHYLAMINE is found on Page 12.\n\n")
          (run_rb ?rulebase "1-BROMO ETHYLBENZENE") (cut)
           (println "***** 1-BROMO ETHYLBENZENE is found on Page 13.\n\n")
          (run_rb ?rulebase "2-AMINO-1-HEXANOL") (cut)
           (println "***** 2-AMINO-1-HEXANOL is found on Page 14.\n\n")
          (run_rb ?rulebase "SERINE") (cut)
           (println "***** SERINE is found on Page 14.\n\n")
          (run_rb ?rulebase "THREONINE") (cut)
           (println "***** THREONINE is found on Page 14.\n\n")
          (run_rb ?rulebase "LACTIC ACID") (cut)
           (println "***** LACTIC ACID is found on Page 15.\n\n")
          (run_rb ?rulebase "GLYCERIC ACID") (cut)
           (println "***** GLYCERIC ACID is found on Page 15.\n\n")
          (run_rb ?rulebase "1,3-DIACETOXY-BUTANE") (cut)
           (println "***** 1,3-DIACETOXY-BUTANE is found on Page 16.\n\n")
    )
  ;*********************************************************************
    ((test1_rb) if
          (println "\n\nWhat is the name of the rule base you want to run tests on?")
          (print "      Please enter it here -> ")
          (getline stdin ?rb)
          (println)
          (:= ?rulebase [strcase ?rb lower])
          (ld_rulebase ?rulebase)
          (ques_let ?rulebase ?descr ?elemdesc ?tratdesc ?itemdesc ?)
          (run_rb ?rulebase "MANDELAMIDE")
           (println "***** MANDELAMIDE is found on Page 4.\n\n")
          (run_rb ?rulebase "PRACTOLOL")
           (println "***** PRACTOLOL is found on Page 4.\n\n")
          (run_rb ?rulebase "2 BUTYL PHENYL CARBAMATE")
           (println "***** 2 BUTYL PHENYL CARBAMATE is found on Page 4.\n\n")
```

```
          (run_rb ?rulebase "D PHENYLALANINE")
           (println "***** D-PHENYLALANINE is found on Page 5.\n\n")
          (run_rb ?rulebase "HISTIDINE")
           (println "***** HISTIDINE is found on Page 5.\n\n")
          (run_rb ?rulebase "2-AMINO-3-PHENYLPROPIONAMIDE")
           (println "***** 2-AMINO-3-PHENYLPROPIONAMIDE is found on Page 5.\n\n")
          (run_rb ?rulebase "TRIADIMEFON")
           (println "***** TRIADIMEFON is found on Page 6.\n\n")
          (run_rb ?rulebase "ACETYLPHENETHOLIDE")
           (println "***** ACETYLPHENETHOLIDE is found on Page 6.\n\n")
          (run_rb ?rulebase "N-(PHENYLETHYL)ACETAMIDE")
           (println "***** N-(PHENYLETHYL)ACETAMIDE is found on Page 6.\n\n")
)
((test2_rb) if
          (println "\n\nWhat is the name of the rule base you want to run test2 on?")
          (print "      Please enter it here -> ")
          (getline stdin ?rb)
          (println)
          (:= ?rulebase (strcase ?rb lower))
          (ld_rulebase ?rulebase)
          (run_rb ?rulebase "N,O-DIBENZOYL2-AMINOBUTANOL")
           (println "***** N,O-DIBENZOYL2-AMINOBUTANOL is found on Page 7.\n\n")
          (run_rb ?rulebase "FENOPROFEN")
           (println "***** FENOPROFEN is found on Page 8.\n\n")
          (run_rb ?rulebase "METHYL-3-BENZOYLOXYPROPANOATE")
           (println "***** METHYL-3-BENZOYLOXYPROPANOATE is found on Page 8.\n\n")
          (run_rb ?rulebase "FLURIPROFEN ME ESTER")
           (println "***** FLURIPROFEN ME ESTER is found on Page 8.\n\n")
          (run_rb ?rulebase "METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE")
           (print "***** METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE is ")
           (println "on Page 9.\n\n")
          (run_rb ?rulebase "METHYL2-PHENYLPROPIONATE")
           (println "***** METHYL2-PHENYLPROPIONATE is found on Page 9.\n\n")
          (run_rb ?rulebase "1 METHYL 3 PHENYL PROPYL BENZOATE")
           (println "***** 1 METHYL 3 PHENYL PROPYL BENZOATE is found on Page 9.\n\n")
)
((test3_rb) if
          (println "\n\nWhat is the name of the rule base you want to run test3 on?")
          (print "      Please enter it here -> ")
          (getline stdin ?rb)
          (println)
          (:= ?rulebase (strcase ?rb lower))
          (ld_rulebase ?rulebase)
          (run_rb ?rulebase "GUALFENESINE")
           (println "***** GUALFENESINE is found on Page 10.\n\n")
          (run_rb ?rulebase "9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE")
           (println "***** 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE is ")
           (println "on Page 9.\n\n")
          (run_rb ?rulebase "1 PHENYL-1,2-ETHANEDIOL")
           (println "***** 1-PHENYL-1,2-ETHANEDIOL is found on Page 10.\n\n")
          (run_rb ?rulebase "PERISOXAL")
           (println "***** PERISOXAL is found on Page 11.\n\n")
          (run_rb ?rulebase "PINDOLOL")
           (println "***** PINDOLOL is found on Page 11.\n\n")
          (run_rb ?rulebase "METOPROLOL")
           (println "***** METOPROLOL is found on Page 11.\n\n")
          (run_rb ?rulebase "CARBINOXAMINE")
           (println "***** CARBINOXAMINE is found on Page 12.\n\n")
          (run_rb ?rulebase "1-PHENYLETHYLAMINE")
           (println "***** 1-PHENYLETHYLAMINE is found on Page 12.\n\n")
)
((test4_rb) if
          (println "\n\nWhat is the name of the rule base you want to run test4 on?")
          (print "      Please enter it here -> ")
          (getline stdin ?rb)
          (println)
          (:= ?rulebase (strcase ?rb lower))
          (ld_rulebase ?rulebase)
          (run_rb ?rulebase "1-BROMO ETHYLBENZENE")
           (println "***** 1-BROMO ETHYLBENZENE is found on Page 13.\n\n")
          (run_rb ?rulebase "2-AMINO-1-HEXANOL")
           (println "***** 2-AMINO-1-HEXANOL is found on Page 14.\n\n")
          (run_rb ?rulebase "SERINE")
           (println "***** SERINE is found on Page 14.\n\n")
          (run_rb ?rulebase "THREONINE")
           (println "***** THREONINE is found on Page 14.\n\n")
          (run_rb ?rulebase "LACTIC ACID")
           (println "***** LACTIC ACID is found on Page 15.\n\n")
          (run_rb ?rulebase "GLYCERIC ACID")
           (println "***** GLYCERIC ACID is found on Page 15.\n\n")
          (run_rb ?rulebase "1,3-DIACETOXY-BUTANE")
           (println "***** 1,3-DIACETOXY-BUTANE is found on Page 16.\n\n")
)
```

```
; (ld_rulebase ?rulebase) - Load the rulebase ?rulebase that contains the
;           PCT rulebase, if it is not already been loaded.
;           Case 1: Determine if the PCT rulebase has already been loaded
;           Case 2:  If Case 1 fails, it hasn't been loaded so load it
  ((ld_rulebase ?rulebase) if
           (current_predicate ?rulebase 3)
           (println "\nThe Rulebase ->" ?rulebase "<- is already loaded.\n\n")
           (cut)
    )
  ((ld_rulebase ?rulebase) if
           (consult ?rulebase)
           (println "\nThe Rulebase ->" ?rulebase "<- has been loaded.\n\n")
           (cut)
    )

; (ask4item ?descr ?elemdesc ?itemdesc ?iname) - For each PCT rulebase,
;           there are 4 descriptions that are allowed for:
;                    ?descr is a general description of the problem
;                    ?elemdesc is a description of what the PCT elements are
;                    ?itemdesc is a description of the PCT Traits
;                    ?iname is a description of the item, if it is used.
;           If the rules in the PCT rulebase deal with a specific item, ?iname
;           must be specified.  Lets take choosing a Chiral Chromatography
;           column for example.  The descriptions would be: general - Chirality,
;           PCT elements - Chiral columns, PCT traits - Chemical properties,
;           item - Molecule.  Here, each time you run the system you want to
;           answer the questions based on a specific molecule for which you
;           want to choose a column.
;           Case 1: No item was provided
;           Case 2: Ask user for item name
; NOTE - If the procedure [check4item] isn't modified for each PCT
;           Rulebase that uses an item, problems could arise if the item
;           name input does not exist.  If this is the case, the
;           determination of any trait value by means of a pre-defined
;           Prolog predicate in [exrule] will always result in the
;           predicate failing.  For example, a predicate searching for a
;           carbonyl group would fail if the molecule name was not supplied,
;           which would satisfy Case 2 of [ask_user] incorrectly.
  ((ask4item ? ? "" ""))
           (println "\n      No item name to ask for.....")
    )
  ((ask4item ?descr ?elemdesc ?itemdesc ?iname) if
           (print "\nIn looking at " ?descr ", trying to determine " ?elemdesc)
           (println ",\n   what " ?itemdesc " would you like to investigate?")
           (print "      Enter the " ?itemdesc " name here --> ")
           (getline stdin ?name)
           (check4mol ?name ?iname)                          ;Make sure molecule exists
           (println "\n")                        ;CAN REMOVE ABOVE LINE - SEE ABOVE NOTE
    )

; (check4mol ?name ?iname) - Make sure that the item name ?name exists.
;           If it does, pass the name back in ?iname.  If not, try inputting
;           it from a file (then determine the smiles string for it).
;           It must be modified for each PCT Rulebase that is used IF the user
;           wants to assure that an item name entered exists.
;           Case 1: The item name exists in the runtime molecule database.
;           Case 2: The item name entered is actually a file name.
;           Case 3: Item name dosn't exist, so fail.
; NOTE - If this procedure isn't modified for each PCT Rulebase that
;           uses an item, problems could arise if the item name input does
;           not exist.  If this is the case, the determination of any trait
;           value by means of a pre-defined Prolog predicate in [exrule] will
;           always result in the predicate failing.
  ((check4mol ?name ?name) if
           (molecule ?name ? ? ?)                            ;Check to see that ?iname exists!
           (print "\n  [" ?name "] currently exists in runtime Molecule ")
             (println "database.....")
           (cut)
    )
  ((check4mol ?name ?iname) if
           (println "\n  Inputting molecule from file [" ?name "].....")
           (ld_mol ?name ?iname)                             ;Load the molecule
           (println "\n  Getting molecule Smiles string for [" ?iname "].....")
           (get_smiles ?iname)                               ; and find its Smiles String
           (cut)
    )
  ((check4mol ?name ?) if
           (println "\n      ERROR - The molecule [" ?name "] doesn't exist.....")
           (cut) (fail)
    )

; (ask_user ?qet_list ?Name ?iname ?Name1) - Ask the user for an answer
;           (yes or no) for each of the traits contained in ?qet_list.  Each
```

```
;          trait consists of a list with three pieces of information: a trait
;          description, and the name for the left and right pole of the trait.
;          ?Name is a description of the problem with a ": " appended.  It is
;          used to construct a list of responses to the trait questions, which
;          is finally returned in ?Namef by Case 1.  ?iname is the item name, if
;          it is present.
;          Case 1: When ?qst list is empty, you are done asking questions.  Pass
;                  back the completed list of answers.
;          Case 2: A predicate for the left-hand pole of the construct exists,
;                  and successfully executes.  The answer is "yes" to the question.
;          Case 3: A predicate for the left-hand pole of the construct exists,
;                  and execution fails.  The answer is "no" to the question.
;          Case 4: A predicate for the right-hand pole of the construct exists,
;                  and successfully executes.  The answer is "no" to the question.
;          Case 5: No predicates for the construct exist.  Ask the user for the
;                  answer to the question.
;  NOTE - Case 4 would tend to indicate that another case would be
;          appropriate, where the right-hand pole of the construct exists but
;          execution fails.  This case was not deemed to be practical.
;             - The duplicated lines in Cases 2&3 could be combine into one
;          case using an "or" statement.
;             - The call to "current_predicate" is there to allow future
;          expansion to linking variables of predicates.  Using a "1" ensures
;          that only one variable is passed, the name of the molecule.  In
;          linking predicates, another case would be developed using a "1",
;          where the first variable would be the name of the fact to assert,
;          the second var would be the value to use, and the third would be
;          the name of the molecule.
  ((ask_user nil ?Name ? ?Namef) if                          ;C1-Done w/questions
    (println "Finished asking questions!!\n")
    (list-cat ?Name ( ) ?Namef)
   )
  ((ask_user (?First.?Rest) ?Name ?item ?Namef) if           ;C2-Quest exists
           (split ?Fir ?Sec ?Third ?First)
;          (println "   Looking for pre-existing " ?Sec " with item " ?item ".")
           (current_predicate ?Sec 1)
;          (println "   Found " ?Sec " predicate for " ?Fir ", will it succeed?")
           (exrule ?Sec ?value ?item)
           (println "   For " ?Fir ", Found that " ?Sec " exists. [" ?value "]\n")
           (list-cat ?Name (A ) ?Name2)
           (assert ((fact ?Fir ?Sec)) )
           (ask_user ?Rest ?Name2 ?item ?Namef)
    )
  ((ask_user (?First.?Rest) ?Name ?item ?Namef) if     ;C3-Quest exists, fails
           (split ?Fir ?Sec ?Third ?First)
;          (println "   Looking for pre-existing " ?Sec ", but having it fail.")
           (current_predicate ?Sec 1)
;          (println "   Found " ?Sec " predicate for " ?Fir ", will it fail?")
           (not (exrule ?Sec ? ?item))
;          (print "   For " ?Fir ", Found " ?Sec " dosn't exist, so it's ")
           (println ?Third ".\n")
           (list-cat ?Name (B ) ?Name2)
           (assert ((fact ?Fir ?Third)) )
           (ask_user ?Rest ?Name2 ?item ?Namef)
    )
  ((ask_user (?First.?Rest) ?Name ?item ?Namef) if           ;C4-Opposite exists
           (split ?Fir ?Sec ?Third ?First)
;          (println "   Looking for pre-existing opposite of " ?Sec ".")
           (current_predicate ?Third 1)
;          (print "      Found " ?Third " (opp) predicate for " ?Fir ", will ")
;          (println "it succeed?")
           (exrule ?Third ?value ?item)
           (println "   For " ?Fir ", Found " ?Third " exists. [" ?value "]\n")
           (list-cat ?Name (B ) ?Name2)
           (assert ((fact ?Fir ?Third)) )
           (ask_user ?Rest ?Name2 ?item ?Namef)
    )
  ((ask_user (?First.?Rest) ?Name ?item ?Namef) if           ;C5-Ask question
           (split ?Fir ?Sec ?Third ?First)
           (println "WHICH OF THE FOLLOWING IS TRUE ABOUT    " ?Fir)
           (println "      A] " ?Sec)
           (println "      B] " ?Third)
           (print "Please type in your response as A/B ->  ")
           (read ?Char)
           (:= ?Ch [strcase ?Char upper])
           (answer ?Ch ?Fir ?Sec ?Third ?Letter)
           (list-cat ?Name ?Letter ?Name2)
           (ask_user ?Rest ?Name2 ?item ?Namef)
    )

; (answer ?ans ?Fir ?Sec ?Third ?Letter) - Asserts facts in the dynamic
;          database according to what the user inputs.  ?ans is passed to the
;          predicate, and is used to choose between the three cases below.
;          ?Fir, ?Sec, ?Third are the trait description, left-hand and right-
;          hand poles respectively.  ?Letter contains the answer that is also
;          passed back to Case 5 of predicate "ask_user".
;          Case 1: Assert the fact using the trait description and the
;                  left-hand pole of the construct
;          Case 2: Assert the fact using the trait description and the
;                  right-hand pole of the construct
;          Case 3: Do not assert fact, don't understand input.
  ((answer "A" ?Fir ?Sec ?Third (A )) if     ; Assert answers in
           (println "  You chose A, which was " ?Sec ".\n")  ; dynamic database
           (assert ((fact ?Fir ?Sec)) )
    )
  ((answer "B" ?Fir ?Sec ?Third (B )) if
           (println "  You chose B, which was " ?Third ".\n")
           (assert ((fact ?Fir ?Third)) )
    )
  ((answer ?Ch ?Fir ?Sec ?Third (X )) if
           (println "  You chose was " ?Ch ".\n")
    )

; (split ?Fir ?Sec ?Third ?qlist) - The information on each trait is
;          provided as a list of lists.  Each individual trait has a description,
;          the left-hand pole name and right-hand pole name, that are contained
;          within a list.  This predicate brakes up the trait informaiton triad
;          of information on an individual trait and passes each back as an
;          argument.
;          Case split: Remove the trait description from the description list
;          Case split2: Remove the left-hand pole from the description list
;          Case split3: Remove the right-hand pole from the description list
  ((split ?Fir ?Sec ?Third (?Fir.?Rest)) if  ; Split up list of questions
           (split2 ?Sec ?Third ?Rest)   )
  ((split2 ?Sec ?Third (?Sec.?Rest)) if
           (split3 ?Third ?Rest)    )
  ((split3 ?Third (?Third.nil)) )

; (exrule ?trat_lp ?value ?item) - Instead of asking the user for the value
;          for a particular PCT trait, see if that trait is already defined
;          as a prolog predicate that can be executed to determine a value.
;          Case 1: No item name is provided, so return the value passed back
;                  by the execution of the trait pole.
;          Case 2: Item is provided, so execute the PCT rule using the item name
;                  and pass back the item name as ?value.
  ((exrule ?trat_pole ?value "") if
;          (println "         Attempting " ?trat_pole ".....")
           (?trat_pole ?value)
;          (println "         Executed " ?trat_pole " and returned " ?value ".")
           (cut)
    )
  ((exrule ?trat_pole ?item ?item) if        ; pass back ?item
;          (println "         Attempting " ?trat_pole " using " ?item ".....")
           (?trat_pole ?item)
;          (println "         Executed " ?trat_pole " using " ?item ".")
    )

; (search_rules ?rulebase ?Num) - Execute all the rules in the rulebase
;          ?rulebase.  ?Num is a counter to keep track of which rule should be
;          executed.
;          Case 1: Execute a rule in ?rulebase
;          Case 2: This will fail when all the rules have been executed.
; NOTE - ?Max will contain the maximum number of PCT rules that can be
;                  generated.  In the event that the PCT grid has a value of "3",
;                  which means not the left or right hand trait pole, there will be
;                  no rule generated.  Hence, there may be cases where there are
;                  LESS than ?Max number of rules.  This will not cause a problem.
  ((search_rules ?rulebase ?Num) if
;     (println "  T E S T I N G    R U L E   " ?Num)
           (testrule ?rulebase ?Num)
           (:= ?Nxt [+ ?Num 1])
           (search_rules ?rulebase ?Nxt)
           )
  ((search_rules ?rulebase ?Num) if    ; Used after all rules are searched
           (maxrules ?Max)
           (> ?Num ?Max)
           (println "End of Rules!\n")
           )

; (testrule ?rulebase ?Num) - Executes the rule number ?Num in ?rulebase
;          and calls compute_cv to calculate the new confidence factor.
;          Case 1: Current rule succeeds, calculate the CF and go on
;          Case 2: Current rule fails, so go on to next rule.  Fail only if
;                  all rules have been executed.
;    NOTE: If the grid value is "3" then a rule will NOT be generated for
;                  it.  This will result in less than "maxrules" number of rules.
```

```
;                    Case 2 will handle the testing of these non-existant rules.
  ((testrule ?rulebase ?Num) if
             (?rulebase ?Num ?Ele ?Cf)
             (compute_cf ?Ele ?Cf)
    )

  ((testrule ? ?Num) if                  ; Used when a rule fails and there are
             (maxrules ?Max)                 ;   still rules to be tested
             (<= ?Num ?Max)
    )

; (compute_cf ?Ele ?Cf) - Recall the old confidence factor associated
;          with the element ?Ele, combine it with the newly discovered ?Cf,
;          retract the old and assert the newly calculated confidence factor.
;          Case 1: Combine two negative ?Cf's (special case)
;          Case 2: Combine the two ?Cf's
;          Case 3: Used the first time there is a ?Cf for ?Ele
  ((compute_cf ?Ele ?Cf) if              ; special case when both
             (< ?Cf 0)                           ;   CF's are negative
             (cf ?Ele ?Old_cf)                   ; Recall current Cf
             (< ?Old_cf 0)
             (:= ?Rel1 (/ (* (+ 100 ?Old_cf) ?Cf) 100))
             (:= ?New_cf (+ ?Old_cf ?Rel1))              ; Calculate new Cf
;     (println "          " ?Rel1 " added to " ?Old_cf)
;     (println "     " ?Ele "  " ?Newcf)
             (retract ((cf ?Ele ?Old_cf)) )                ; Delete old Cf
             (assert ((cf ?Ele ?New_cf)) )                 ; Save new Cf
    )
   ((compute_cf ?Ele ?Cf) if
             (cf ?Ele ?Old_cf)                           ; Recall current Cf
             (:= ?Rel1 (/ (* (- 100 ?Old_cf) ?Cf) 100))
             (:= ?New_cf (+ ?Old_cf ?Rel1))              ; Calculate new Cf
;     (println "          " ?Rel1 " added to " ?Old_cf)
             (retract ((cf ?Ele ?Old_cf)) )                ; Delete old Cf
             (assert ((cf ?Ele ?New_cf)) )                 ; Save new Cf
    )
  ((compute_cf ?Ele ?Cf) if          ; used first time element has
             (assert ((cf ?Ele ?Cf)) )             ;  a cf to put into data base
    )

; (printresults) - retract and print out the largest element and its
;          confidence factor, retracting and printing the rest by recursively
;          calling itself.
;          Case 1: Retract any cf to make sure there is at least one, and call
;                       findlargest to see if it is the largest, print out and retract
;                       the largest cf.
;          Case 2: All elements and cf's printed out
  ((printresults) if
             (cf ?Ele ?Cf)                                 ; take any cf and retract it
             (retract ((cf ?Ele ?Cf)) )
             (findlargest ?Ele ?Cf ?Hele ?Hcf)             ; find the largest
             (assert ((cf ?Ele ?Cf)) )
             (println ?Hele "   WITH A PROBABILITY OF    " (ceil ?Hcf))
             (retract ((cf ?Hele ?Hcf)) )                  ; print it out and retract it
             (cut)
             (printresults)
    )
  ((printresults) if (println)   )

; (findlargest ?Ele ?Cf ?Hele ?Hcf) - Search through the entire dynamic
;          database cf to see if ?Ele has the largest confidence factor ?Cf.
;          When it finds the largest, return it in ?Hele and ?Hcf.  It uses
;          the following logic:  Look at two CF's, retract the smaller of the
;          two, recursively call itself to find the smallest one.  When it finds
;          the smallest, pass it back through in ?Hele & ?Hcf, and re-assert
;          the other remaining CF's.
;          Case 1: Gets a new CF, and if the new CF is greater than the original
;                       one, retract the new one and see if the new one is the largest.
;          Case 2: Gets a new CF, and if the new CF is less than or equal to
;                       the original one, retract the new one and see if the original
;                       one is the largest.
;          Case 3: Found the largest cf, so return it in ?Hele and ?Hcf.
  ((findlargest ? ?Cf ?Hele ?Hcf) if
             (cf ?Nele ?Ncf)                               ; get another cf
             (< ?Cf ?Ncf)                                  ; is it GT the cf passed it?
             (retract ((cf ?Nele ?Ncf)) )                  ; if so retract it and see if
             (findlargest ?Nele ?Ncf ?Hele ?Hcf); the new one is the largest
             (assert ((cf ?Nele ?Ncf)) )
    )
  ((findlargest ?Ele ?Cf ?Hele ?Hcf) if
             (cf ?Nele ?Ncf)                               ; get another cf
             (>= ?Cf ?Ncf)                                 ; is it LTE the cf passed it?
             (retract ((cf ?Nele ?Ncf)) )                  ; if so retract it and see if
             (findlargest ?Ele ?Cf ?Hele ?Hcf) ; original is larger than the rest
             (assert ((cf ?Nele ?Ncf)) )
    )
  ((findlargest ?Ele ?Cf ?Hele ?Hcf) if
             (= ?Hele ?Ele)
             (= ?Hcf ?Cf)
    )

; (deletefacts ?get_list) - Retract and print out and all of the
;          dynamic database that contains responses for the trait questions.
;          Case 1: All the trait answers have been printed out.
;          Case 2: Print the answer for the current trait, calling itself
;                       recursively to print them all out.
;          Case 3: If no fact is found for the current trait, none was entered.
  ((deletefacts nil) if
;          (println "Finished running.")
    )
  ((deletefacts (?First.?Rest)) if
             (split ?Fir ? ? ?First)
             (fact ?Fir ?Ans)
             (println "       For " ?Fir ", you chose " ?Ans ".")
             (retract ((fact ?Fir ?Ans)) )
             (deletefacts ?Rest)
    )
  ((deletefacts (?First.?Rest)) if
             (split ?Fir ? ? ?First)
             (println "For " ?Fir ", you did not choose a response.")
             (deletefacts ?Rest)
    )

; (list-cat ?lista ?listb ?listout) - concatenate second list to end of the
;          first list.  From file BASEPREDS.HC.  The name has been changed, and
;          it is not needed when combine with the functional group program FG.HC.
             ((list-cat nil ?list ?list) if (cut) )
             ((list-cat ?list nil ?list) if (cut) )
             ((list-cat (?h . ?t) ?sec (?h . ?ans)) if
                          (cut)
                          (list-cat ?t ?sec ?ans)  )

;  ***END OF assert
)
```

TurboProlog Inference Engin

```
code=2000
domains
  namelist = nlist*
  nlist = name*
  name = symbol
  Ch = char
  rnum = integer
  certf = real

database
  fact(name,name)
  cf(name,certf)

predicates
  run
  ques_list(namelist)
  ask_user(namelist,name,name)
  answer(char,name,name,name,name)
  split(name,name,name,nlist)
  split2(name,name,nlist)
  split3(name,nlist)
  search_rules(rnum)
  testrule(rnum,name)
  compute_cf(name,certf,certf)
  printresults
  deletefacts(namelist)
  findlargest(name,certf,name,certf)
  maxrules(rnum)
  rule(rnum,name,name,certf)

goal
  run.

clauses
  run  :-
     readdevice(stdin),writedevice(stdout),
     write("What is the name of this item you are classifying?"),nl,
     write("      Please enter it here -> "),
```

```
    readln(Name),nl,
    concat(Name," : ",Name2),
    ques_lst(Namelist),
    ask_user(Namelist,Name2,Namef),
    search_rules(1),
    !,
    write("The following is the data for ",Namef),nl,nl,
    printresults,
    deletefacts(Namelist).

ask_user([],Name,Namef) :-
  write("Finished asking questions!!"),nl,nl,
  concat(Name," ",Namef).

ask_user([First|Rest],Name,Namef) :- /* Ask all questions */
    split(Fir,Sec,Third,First),
    write ("WHICH OF THE FOLLOWING IS TRUE ABOUT    "),
    write (Fir),nl,
    write("      A] ",Sec),nl,
    write("      B] ",Third),nl,
    write("Please type in your response as A/B ->  "),
    readchar(Ch),readchar(_),write(Ch),nl,nl,
    answer(Ch,Fir,Sec,Third,Letter),
    concat(Name,Letter,Name2),
    ask_user(Rest,Name2,Namef).

answer(Ch,Fir,Sec,Third,Letter) :-     /* Assert answers in dynamic database */
    Ch='A',write("You chose A, which was ",Sec),nl,nl,
    Letter = "A ",
    asserta(fact(Fir,Sec))
   or
    Ch='a',write("You chose A, which was ",Sec),nl,nl,
    Letter = "A ",
    asserta(fact(Fir,Sec))
   or
    Ch='B',write("You chose B, which was ",Third),nl,nl,
    Letter = "B ",
    asserta(fact(Fir,Third))
   or
    Ch='b',write("You chose B, which was ",Third),nl,nl,
    Letter = "B ",
    asserta(fact(Fir,Third))
   or
    write("Your choice was ",Ch),nl,nl,
    Letter = "X ".

split(Fir,Sec,Third,[Fir|Rest]) :-   /* Split up list of questions */
    split2(Sec,Third,Rest).
split2(Sec,Third,[Sec|Rest]) :-
    split3(Third,Rest).
split3(Third,[Third|[]]).

search_rules(Num) :-
     write("  T E S T I N G    R U L E   ",Num),nl,
     testrule(Num,element),
     Nxt = Num + 1,
     search_rules(Nxt)
    or      /* Used after all rules have been searched */
     maxrules(Max),
     Num > Max,
     nl,write("End of Rules "),nl,nl.

testrule(Num,element) :-
    rule(Num,element,Ele,Cf),
    compute_cf(Ele,Cf,Newcf),
    write(Ele,"     ",Newcf),nl
   or      /* Used when a rule fails and there are */
    maxrules(Max),    /*  still rules left to be tested */
    Num <= Max.

compute_cf(Ele,Cf,New_cf) :- /* special case when both CF's are negative */
    Cf < 0,
    cf(Ele,Old_cf),                     /* Recall current Cf */
    Old_cf < 0,
    Rel1 = ((((100 + Old_cf) * Cf) / 100),
    New_cf = Old_cf + Rel1,             /* Calculate new Cf */
    write(Rel1," added to ",Old_cf),nl,
    retract( cf(Ele,Old_cf) ),          /* Delete old Cf */
    asserta( cf(Ele,New_cf) ).          /* Save new Cf */

compute_cf(Ele,Cf,New_cf) :-
    cf(Ele,Old_cf),                     /* Recall current Cf */
    Rel1 = ((((100 - Old_cf) * Cf) / 100),
    New_cf = Old_cf + Rel1,             /* Calculate new Cf */
    write(Rel1," added to ",Old_cf),nl,
    retract( cf(Ele,Old_cf) ),          /* Delete old Cf */
    asserta( cf(Ele,New_cf) ).          /* Save new Cf */

compute_cf(Ele,Cf,Cf) :- /* used first time element has */
    asserta( cf(Ele,Cf) ).    /*  a cf to put into data base */

printresults :-
    cf(Ele,Cf),
    retract( cf(Ele,Cf) ),
    findlargest(Ele,Cf,Hele,Hcf),
    assert( cf(Ele,Cf) ),
    write(Hele,"   WITH A PROBABILITY OF    ",Hcf),nl,
    retract( cf(Hele,Hcf) ),
    printresults
   or
    nl.

findlargest(_,Cf,Hele,Hcf) :-
    cf(Nele,Ncf),
    Cf < Ncf,
    retract( cf(Nele,Ncf) ),
    findlargest(Nele,Ncf,Hele,Hcf),
    assert( cf(Nele,Ncf) ).

findlargest(Ele,Cf,Hele,Hcf) :-
    cf(Nele,Ncf),
    Cf >= Ncf,
    retract( cf(Nele,Ncf) ),
    findlargest(Ele,Cf,Hele,Hcf),
    assert( cf(Nele,Ncf) ).

findlargest(Ele,Cf,Hele,Hcf) :-
    Hele = Ele,
    Hcf = Cf.

deletefacts ([]) :-
    nl,write("Finished running."),nl.

deletefacts([First|Rest]) :-
    split(Fir,_,_,First),
    fact(Fir,Ans),
    write("For ",Fir,", you chose ",Ans,"."),nl,
    retract( fact(Fir,Ans) ),
    deletefacts(Rest)
   or
    split(Fir,_,_,First),
    write("For ",Fir,", you did not choose a response."),nl,
    deletefacts(Rest).

/* generated using PCT */

ques_lst([
     [aromatic, has_benz, no_benz],
     [carbonyl, has_cdo, no_cdo],
     [alchol, has_oh, no_oh],
     [tert_nitrogen, has_n3, no_n3],
     [primary_amine, has_np, no_np],
     [carboxylic_acid, has_co2h, no_co2h]      ]).

maxrules ( 108).

rule (1, element, p4, 43) :- fact(aromatic, has_benz).
rule (2, element, p4, -43) :- fact(aromatic, no_benz).
rule (3, element, p5_7, 43) :- fact(aromatic, has_benz).
rule (4, element, p5_7, -43) :- fact(aromatic, no_benz).
rule (5, element, p8_9, 43) :- fact(aromatic, has_benz).
rule (6, element, p8_9, -43) :- fact(aromatic, no_benz).
rule (7, element, p10_11, 43) :- fact(aromatic, has_benz).
rule (8, element, p10_11, -43) :- fact(aromatic, no_benz).
rule (9, element, p12, 43) :- fact(aromatic, has_benz).
rule (10, element, p12, -43) :- fact(aromatic, no_benz).
rule (11, element, p13, 43) :- fact(aromatic, has_benz).
rule (12, element, p13, -43) :- fact(aromatic, no_benz).
rule (13, element, p14, 43) :- fact(aromatic, has_benz).
rule (14, element, p14, 43) :- fact(aromatic, no_benz).
rule (15, element, p15, -43) :- fact(aromatic, has_benz).
rule (16, element, p15, 43) :- fact(aromatic, no_benz).
rule (17, element, p16, -43) :- fact(aromatic, has_benz).
rule (18, element, p16, 43) :- fact(aromatic, no_benz).
rule (19, element, p4, 43) :- fact(carbonyl, has_cdo).
rule (20, element, p4, -43) :- fact(carbonyl, no_cdo).
```

```
rule (21, element, p5_7, 43) :- fact(carbonyl, has_cdo).
rule (22, element, p5_7, -43) :- fact(carbonyl, no_cdo).
rule (23, element, p8_9, 43) :- fact(carbonyl, has_cdo).
rule (24, element, p8_9, -43) :- fact(carbonyl, no_cdo).
rule (25, element, p10_11, -43) :- fact(carbonyl, has_cdo).
rule (26, element, p10_11, 43) :- fact(carbonyl, no_cdo).
rule (27, element, p12, -43) :- fact(carbonyl, has_cdo).
rule (28, element, p12, 43) :- fact(carbonyl, no_cdo).
rule (29, element, p13, -43) :- fact(carbonyl, has_cdo).
rule (30, element, p13, 43) :- fact(carbonyl, no_cdo).
rule (31, element, p15, 43) :- fact(carbonyl, has_cdo).
rule (32, element, p15, -43) :- fact(carbonyl, no_cdo).
rule (33, element, p4, 43) :- fact(alchol, has_oh).
rule (34, element, p4, -43) :- fact(alchol, no_oh).
rule (35, element, p5_7, -43) :- fact(alchol, has_oh).
rule (36, element, p5_7, 43) :- fact(alchol, no_oh).
rule (37, element, p8_9, -43) :- fact(alchol, has_oh).
rule (38, element, p8_9, 43) :- fact(alchol, no_oh).
rule (39, element, p10_11, 43) :- fact(alchol, has_oh).
rule (40, element, p10_11, -43) :- fact(alchol, no_oh).
rule (41, element, p12, -43) :- fact(alchol, has_oh).
rule (42, element, p12, 43) :- fact(alchol, no_oh).
rule (43, element, p13, -43) :- fact(alchol, has_oh).
rule (44, element, p13, 43) :- fact(alchol, no_oh).
rule (45, element, p15, 43) :- fact(alchol, has_oh).
rule (46, element, p15, -43) :- fact(alchol, no_oh).
rule (47, element, p5_7, 43) :- fact(tert_nitrogen, has_n3).
rule (48, element, p5_7, -43) :- fact(tert_nitrogen, no_n3).
rule (49, element, p8_9, -43) :- fact(tert_nitrogen, has_n3).
rule (50, element, p8_9, 43) :- fact(tert_nitrogen, no_n3).
rule (51, element, p12, 43) :- fact(tert_nitrogen, has_n3).
rule (52, element, p12, -43) :- fact(tert_nitrogen, no_n3).
rule (53, element, p13, -43) :- fact(tert_nitrogen, has_n3).
rule (54, element, p13, 43) :- fact(tert_nitrogen, no_n3).
rule (55, element, p14, 43) :- fact(tert_nitrogen, has_n3).
rule (56, element, p14, -43) :- fact(tert_nitrogen, no_n3).
rule (57, element, p14, 43) :- fact(primary_amine, has_np).
rule (58, element, p14, -43) :- fact(primary_amine, no_np).
rule (59, element, p15, -43) :- fact(primary_amine, has_np).
rule (60, element, p15, 43) :- fact(primary_amine, no_np).
rule (61, element, p16, -43) :- fact(primary_amine, has_np).
rule (62, element, p16, 43) :- fact(primary_amine, no_np).
rule (63, element, p15, 43) :- fact(carboxylic_acid, has_co2h).
rule (64, element, p15, -43) :- fact(carboxylic_acid, no_co2h).
rule (65, element, p16, -43) :- fact(carboxylic_acid, has_co2h).
rule (66, element, p16, 43) :- fact(carboxylic_acid, no_co2h).
```

APPENDIX F: CHESS Source Code

Appendix F contains the Prolog code that makes up CHESS. It is split into 9 modules: RES.HC, MOL.HC, FG.HC, RING.HC, SMILES.HC, SPARC.HC, NEW.HC, and CHIRULE.HC. Consult Table XII for a description of each of these modules. Section VII.D.1 discusses the details on how to load and run CHESS, while Section V.A goes through this process using CHIRULE as an example.

```
;                              **** File RR8.HC ****                    February 26, 1993
(consult new.hc)
(consult fg.hc)                                (consult chem.hc)
(consult ring.hc)                              (consult mol.hc)
(consult smiles.hc)                            (consult sparc.hc)
(consult chirule.hc)           (consult key.hc)

dynamic molecule               ;MOL file info, name, Smiles String, atom & bond lists
dynamic molfrag                ;Molecules Smiles String, and 4 fragments
dynamic fg-valu                ;Fragment, property, and fragments value
dynamic chiral-col             ;Molecules Smiles String, Column used, and reference
dynamic fvalu-range            ;Runtime only, Fragment, property, delta, and value of box
dynamic fragmatch              ;Runtime only, Fragment matching info
dynamic column-sugg            ;Runtime only, Matching molecule SS and match value

; (MOLECULE KNOWLEDGE BASE [MOL.KB] - Contains the information read in from
;             the mol file generated by MDL's MACCS program.  It consists of
;             the fact name "molecule", the name of the molecule, the SMILES
;             string, an atom list and a bond list.  The runtime KB is built
;             by the predicate (load_mol_kb) and added to by (load_mol), while
;             the static KB is built by the predicate (save_mol_kb) and added
;             to by the predicate (save_mol ?name).  Example:
;                             ((molecule molecule-name smiles-string atom-list bond-list))
;                             ((molecule PHENYLGLYCINE "c(ccc1C(N)C(=O)O)cc1"
;                                       ( (atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0)
;                                         (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 O 0)
;                                         (atom 9 O 0) (atom 10 C 3) (atom 11 N 0)           )
;                                       ( (bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2)
;                                         (bond 5 6 1) (bond 6 1 2) (bond 7 8 2) (bond 7 9 1)
;                                         (bond 7 10 1) (bond 10 4 1) (bond 10 11 1)         )
;                             ))

; (SMILES STRING MOLECULE FRAGMENT KNOWLEDGE BASE [MOL-FRAG.KB] - Contains
;             the Smiles string for the entire chiral molecule, and its 4
;             fragments generated by splitting the molecule up around the
;             chiral center.  It consists of the fact name "molfrag", the Smiles
;             string of the molecule, and the Smiles strings of the 4 fragments
;             in order of decreasing number of atoms contained by each.  The
;             runtime "molfrag" KB is built by the predicate (load_molfrag_kb)
;             and added to by (getall_smiles ?name), while the static "molfrag" KB
;             is added to by (saveall_smiles ?name).
;                             ((molfrag MOLSS fgmt1ss fgmt2ss fgmt3ss fgmt4ss))
;                             ((molfrag "c(ccc1C(N)C(=O)O)cc1" "c(ccc1)cc1" O=CO N ""))
; data:       ((molecule PHENYLGLYCINE "c(ccc1C(N)C(=O)O)cc1" ?atlist ?bdlist))

; (SPARC PROPERTY FRAGMENT VALUE KNOWLEDGE BASES [FV-POL.KB, FV-ALPHA.KB, FV-BETA.KB]
;             Contains the values generated by SPARC for the property's listed for
;             each fragment.  It consists of the fact name "fg-valu", the Smiles
;             string of the fragment, the propery abbreviation used by SPARC, and
;             the property values for the fragment.  The runtime fg-valu KB is built
;             by the predicate (load_value_kb), and added to by (getall_sparc ?).  The
;             static KB is built by the predicate (getall_sparc) which calls
;             (savef value).  Values are reported up to and including the 5th
;             decimal place.
;  SPARC Property Abbreviations:
;             pol - Average Molecular Polarizability
;             alpha - Taft-Kamlet approach to proton donating H-bonding
;             beta - Taft-Kamlet approach to proton accepting H-bonding
;             Others - sol, vp, hv, dist, bp, vol, ind.
;                             ((fg-valu FRAGSS property fgmt-valu))
;                             ((fg-valu "c(ccc1)cc1" pol 1.03565))      ((fg-valu N pol 0.13185))
;                             ((fg-valu "c(ccc1)cc1" alpha 0))                ((fg-valu N alpha 0))
;                             ((fg-valu "c(ccc1)cc1" beta 0.12))              ((fg-valu N beta 0.36))
;                             ((fg-valu O=CO pol 0.34022))
;                             ((fg-valu O=CO alpha 0.5952))
;                             ((fg-valu O=CO beta 0.4379))
; data:       ((molecule PHENYLGLYCINE "c(ccc1C(N)C(=O)O)cc1" ?atlist ?bdlist))
; data:       ((molfrag "c(ccc1C(N)C(=O)O)cc1" "c(ccc1)cc1" O=CO N ""))

; (CHIRAL COLUMN KNOWLEDGE BASE [CHIRLCOL.KB] - Consists of the fact name
;             "chiral-col", the smiles string for the molecule, the column or
;             columns used to separate the enantiomers, and the spectra reference
;             (a page number and a "D" for a column from the Daicel catalog).
;             The runtime "chiral-col" KB is built by the predicate "load_chiral_kb"
;             and added to by "getinfo_chiral ?", while the static KB is only added
;             to by "getinfo chiral ?".
;  NOTE: Compounds that could be separated using different columns
;             have individual entries for each column.  Also, other information could
;             be entered here, like eluent, flow rate, detection, and temperature.
;                             ((chiral molecule-smilesstring chiral-column spectra-reference))
;                             ((chiral "c(ccc1C(N)C(=O)O)cc1" "crownpak CR" "Daicel p34))
; data:       ((molecule PHENYLGLYCINE "c(ccc1C(N)C(=O)O)cc1" ?atlist ?bdlist))
; data:       ((molfrag "c(ccc1C(N)C(=O)O)cc1" "c(ccc1)cc1" O=CO N ""))

; (FRAGMENT VALUE RANGE KNOWLEDGE BASE [runtime only] - Consists of the fact name
;             "fvalu-range", the fragment smiles string, the name of the property
;             (pol, alpha, beta), the delta value used to generate the value, and
;             the value itself.  The purpose is to place each fragment into boxes,
;             so that they can be compared to other fragments to come up with
;             "similar" fragments.  Two entries in "fvalu-range" are generated for
;             each value in the fragment knowledge base "fg-valu".
;             Generated by the predicate (gen_fvrange_kb ?property ?delta) from
;             the fragment value knowledge base "fg-valu":
;                             (((fvalu-range ?fragment ?property ?delta ?value))
;                             (((fvalu-range "c(ccc1)cc1" pol 10 1)))
;                                                 (((fvalu-range "c(ccc1)cc1" pol 10 10)))
;                             (((fvalu-range O=CO pol 10 4))                  (((fvalu-range O=CO pol 10 3)))
;                             (((fvalu-range N pol 10 3)))                    (((fvalu-range N pol 10 2)))
; data:       ((molecule PHENYLGLYCINE "c(ccc1C(N)C(=O)O)cc1" ?atlist ?bdlist))
; data:       ((molfrag "c(ccc1C(N)C(=O)O)cc1" "c(ccc1)cc1" O=CO N ""))
; data:       ((fg-valu "c(ccc1)cc1" pol 1.03565))      ((fg-valu N pol 0.13185))
; data:       ((fg-valu O=CO pol 0.34022))

; (FRAGMENT MATCH KNOWLEDGE BASE [runtime only] - Consists of the fact name
;             "fragmatch", the Smiles String of the molecule of interest, the Smiles
;             String of the matching molecule, the number of fragments that were
;             matched (i.e. four minus the number of identical fragments), the
;             fragment in the molecule of interest and its match in the matching
;             molecule, the property used to match the fragments, and
;             Generated by predicate (psimilar ?frag ?fl1 ?fl2 ?delta)
;                             (((fragmatch ?sss ?ssb ?nmatch ?frag ?frag2 ?property ?delta ?diff))

; (COLUMN SUGGESTION KNOWLEDGE BASE [runtime only] - Consists of the fact name
;             "column-sugg", the Smiles String of the molecule of interest, the
;             Smiles String of the molecule that is considered to be a "match", and
;             a numerical value representing the strength of the match.  Lower match
;             numbers mean a CLOSER MATCH, higher match numbers are not as good.
;             Generated by predicate (assert_colsugg ? ? ?), which is called
;             by (calc_diff).
;                             ((column-sugg ?sss ?ssb ?value))

;                                                                  ***** TO-DO LIST & NOTES *****
; 5-92:Should make both a test and a research MOL.KB.
; 5-92:add more molfile symbols to (absubscript ? ? ? ?)
; 5-92:Should POH or SOH count for having "has_OxH" succeeding?
; 5-92:Should Esters specifically exclude anhydrides?
;

(assert

; (area) - clear the environment and load all predicates again
                ((area) if
                                (clearenv)                      (consult res.hc)
                                ;               (abolish load_all_kb)       (abolish load_mol_kb)
                                ;               (abolish load_molfrag_kb)   (abolish load_valu_kb)
                                ;               (abolish firstonlist)       (abolish seconlist)
                                ;               (abolish lastonlist)        (abolish member)
                                ;               (abolish listcat)                           (abolish listsiz)
                                ;               (abolish subsetof)
                                ;               (abolish area)
                )

; (load_all_kb) - Load all of the knowledge bases.  Current knowledge bases include:
;             molecule [MOL.KB] - contains molecule name, ss, bond and atom lists
;             molfrag [MOL-FRAG.KB] - has molecule ss and 4 fgmts, largest fgmt first
;             molvalu [MOL-VALU.KB] - has molecule ss, property, and values for 4 fgmts
;                             in the same order as "molfrag" above
;             chiral [CHIRLCOL.KB] - has molecule ss, the column name, and a reference
                ((load_all_kb) if
                                (load_mol_kb)
                                (load_molfrag_kb)
                                (load_value_kb)
                                (load_chiral_kb)
                )

; (load_mol_kb) - Load the Molecule Knowledge base
                ((load_mol_kb) if
                                (abolish molecule)
                                (loadkb molecule mol.kb YYNN AALL)
                ;               (loadfacts molecule mol.kb AALL)
                )

; (load_molfrag_kb) - Load the molecular fragment knowledge base.  It contains the
;             smiles string of the molecule, and the 4 fragments around the chiral
;             carbon atom.  The first fragment has the largest number of atoms,
;             followed by the next largest, etc.
                ((load_molfrag_kb) if
                                (abolish molfrag)
                                (loadkb molfrag "mol-frag.kb" YNNNN AAAAA)
                ;               (loadfacts molfrag "mol-frag.kb" AAAA)
                )


; (load_value_kb) - Load the molecular fragment value knowledge base.  Each
;             property is stored in a separate file, and contains the fragments
;             SMILES string, the property abbreviation, and a value.
                ((load_value_kb) if
                                (abolish fg-valu)
                                (loadkb fg-valu "fv-pol.kb" YYN AAN)
                                (loadkb fg-valu "fv-alpha.kb" YYN AAN)
                                (loadkb fg-valu "fv-beta.kb" YYN AAN)
                                (loadkb fg-valu "fv-vol.kb" YYN AAN)
                )
```

```
; (load_chiral_kb) - Load the Chiral Column knowledge base.  It contains the Smiles
;             string of the molecule, the column name, and a reference.
                ((load_chiral_kb) if
                                (abolish chiral-col)
                                (loadkb chiral_col chirlcol.kb YNN AAA)
                ;               (loadfacts chiral-col chirlcol.kb AAA)
                )

; (gen_fvrange_kb ?property ?normlz ?delta) - Generate the runtime fragment
;             value range knowledge base for the property ?property at the integral ?delta.
;             Use fail to do this for all of the fragment values in the
;             knowledge base *fq valu ? ? ?*.  Both variables MUST be specified.
; NOTE - In order to find matches at the highest delta possible, the
;             higher value deltas must be generated FIRST before lower value deltas.
                ((gen_fvrange_kb ?property ?normlz ?delta) if
                                (fq-valu ?fragment ?property ?x)
                                (:= ?x1 [* [/ ?x ?normlz] ?delta])
                                (:= ?x2 [ceil ?x1])
                                (:= ?x3 [floor ?x1])
                ;               (print "      [fvalu-range " ?fragment " " ?property " " ?delta)
                ;                               (println " " ?x2 " " ?x3 "].")
                                (assert ((fvalu-range ?fragment ?property ?delta ?x2))              )
                                (not (= ?x2 ?x3))   ; only assert one value if they are the same
                                (assert ((fvalu-range ?fragment ?property ?delta ?x3))              )
                                (fail)
                )

; (gen_all_range ?delta) - Generate the runtime fragment value range knowledge base
;             for all properties at the integral ?delta.  The use of "not"s are
;             required so that predicate "gen_fvrange_kb" will succeed.  See NOTE
;             under predicate "gen_fvrange_kb" about deltas used.
                ((gen_all_range ?delta) if
                                (not (gen_fvrange_kb pol 1 ?delta))                 ;(println " ")
                                (not (gen_fvrange_kb alpha 1 ?delta))               ;(println " ")
                                (not (gen_fvrange_kb beta 1 ?delta))                ;(println " ")
                                (not (gen_fvrange_kb vol 100 ?delta))    ;<- Should be 100
                )

; (                               **** File EXS.HC ****                    February 26, 1993
;                                                                   General Prolog predicates..

; (firstonlist ?list ?fst) - Returns the first element of the list ?list
;             in ?fst.
                ((firstonlist (?fst . ?rst) ?fst) if
                ;               (println "      First on list is " ?fst ", rest is " ?rst ".")
                                (cut)
                )

; (secondlist ?list ?sec) - Returns the second element of the list ?list
;             in ?sec
                ((secondlist (?fst . ?rst) ?sec) if
                                (firstonlist ?rst ?sec)
                                (cut)
                )

; (lastonlist ?list ?lst) - Returns the last member of list ?list in ?lst
;             Case 1 : At last member - return result
;             Case 2 : Not at last member - remove head and call recursively
                ((lastonlist (?lst . nil) ?lst) if
                ;               (println "      Last on list is " ?lst ".")
                                (cut)
                )
                ((lastonlist (?fst . ?rst) ?lst) if
                ;               (println "      Rest of the list is " ?rst ".")
                                (lastonlist ?rst ?lst)
                )

; (member ?atm ?lst) - Check if atom ?atm is contained in the list ?lst.
;             From file BASEPRRDS.HC.
;             Case 1 : Head atom on list matches!  Return.
;             Case 2 : Head atom doesn't match.  Recursively call with the rest of
;                      the list ?tail.
                ((member ?head (?head.?tail)) if
                ;               (println "      Matched!  Member is : " ?head)
                                (cut)
                )
                ((member ?element (?head.?tail)) if
                ;               (println "      Member is : " ?element ".  List is : " ?head ".")
                                (member ?element ?tail)
                )

; (deletemember ?h ?t2 ?t3) - Remove ?h from list ?t2, and return the
;             resulting list in ?t3.  NOTE that predicate fails if ?h is not found
;             in list ?t2 (see removedups case 1 & 4 for further details).
;             Case 1 : Match found with ?h, return rest of list in ?t3.
;             Case 2 : No match found with head atom on list, so call deletemember
;                      recursively with ?t, adding ?h to the returned list as the
;                      predicate recursively comes out.
                ((deletemember ?h (?h.?t) ?t) if        (cut))
                ((deletemember ?x (?h.?t) (?h.?t2)) if
                                (deletemember ?x ?t ?t2)   )

; (listcat ?list1 ?list2 ?listout) - concatenate second list to end of the
;             first list.  From file BASEPRRDS.HC.
                ((listcat nil ?list ?list) if (cut) )
                ((listcat ?list nil ?list) if (cut) )
                ((listcat (?h . ?t) ?sec (?h . ?ans)) if
                                (cut)
                                (listcat ?t ?sec ?ans) )

; (listsiz ?list ?siz) - Determine the number of atoms in the list ?list.
;             The number is calculated as the routines recursively comes out after
;             finding the end of the list in Case 2 and returned in ?siz.
;             Case 1: Split off the first item of the list and recursively call
;                     itself to process the remainding of the list.  When the size
;                     is passed back to it, add one and return.
;             Case 2: Final case when the entire list has been counted.  Return Zero
;                     as the current size to recursively count the number of members.
                ((listsiz (?.?rst) ?siz) if
                                (listsiz ?rst ?size)
                                (:= ?siz [+ ?size 1])       )
                ((listsiz nil 0))

; (subsetof ?rlist ?flist) - Determine of all the members in ?rlist are also
;             members of ?flist.   Cut used to prevent backtracking.
;             Case 1: Split off the first member of the list, see if it is a member
;                     of ?flist, recursively call itself with the rest of the list.
;             Case 2: When all members of ?rlist have been looked at.
                ((subsetof (?fst.?rst) ?flist) if
                                (member ?fst ?flist)
                                (cut)
                                (subsetof ?rst ?flist)     )
                ((subsetof nil ?))

;   ***END OF assert
)


;                               **** File MOL.HC ****                    February 26, 1993
(assert

; (amol) - abolish all predicates in mol.hc so you can try again
                ((amol) if
                                (abolish save_mol_db)                   (abolish add_mol_db)
                                (abolish save_mol)                                (abolish load_mol)
                                (abolish ld_mol)                                  (abolish check_molname)
                                (abolish check_molfrag)                 (abolish fragment_mol)
                                (abolish frag_lst)                                (abolish order_fraglst)
                                (abolish save_frag_mol)                 (abolish renum_fl)
                                (abolish new_atlist)                    (abolish new_bdlist)
                                (abolish atoml)                                   (abolish bondl)
                                (abolish molname)
                                (abolish amol)                                    (consult mol.hc)
                )

; (save_mol_db) - Save all molecules to a new Molecule Knowledge base MOL.DB.
;             Save the old DB to 'MOL.DB.SAV' before starting.
;             Case 1: Write all molecules to the knowledge base.  Use fail to get
;                     all of the molecules.
;             Case 2: Used after all molecules have been written to the knowledge base
; NOTE - This will copy the current molecule knowledge base to MOL.DB.SAV,
;             and proceed saving all molecules to MOL.DB
; PROBLEM - Writing all of the molecule info on one line generates a
;             VERY long line of information.  This may have to be changed.
                ((save_mol_db) if
                                (sys "COPY MOL.DB MOL.DB.SAV")
                                (open mol.db write mol_db)
                                (molecule ?name ?smstr ?atlist ?bdlist)
                                (writeterm mol_db ?name)
                                                (write mol_db " ")
                                                (writeterm mol_db ?smstr)
                                                (writeln mol_db " " ?atlist " " ?bdlist)
                                (println "     Saved molecule " ?name " to Molecule Knowledge base... ")
                                (fail)
                )
                ((save_mol_db) if
                                (println "\n*****  All Molecules saved to Molecule Knowledge base.\n")
                                (close mol_db)
                )

; (add_mol_db) - Input the molecule file name to be added to the
;             Molecule Knowledge base, input the mol file (ld_mol), get the smiles
;             string and add it to the runtime knowledge base (get_smiles), and
;             save it to the permanent molecule knowledge base (save_mol).
                ((add_mol_db) if
        (print "File name for Molecule File: ")
        (getline stdin ?f)
        (ld_mol ?f ?name)
                                (get_smiles ?name)
                                (save_mol ?name)
                (println "\n    Finished adding " ?name " to the Molecule Knowledge base.")
                )

; (save_mol ?name) - Save the molecule ?name to the Molecule Knowledge base.
```

```
;             it ANDUMBO that you want to save the molecule to the Molecule
;             knowledge base, and does NOT DO ANY CHECKING.....
; NOTE - If you don't supply a name with this predicate, you could save
;             ALL CURRENTLY DEFINED MOLECULES!!!   BE CAREFUL!!!
              ((save_mol ?name) if
                            (molecule ?name ?smstr ?atlist ?bdlist)
                            (println * \nSaving * ?name * to the Molecule Knowledge base... *)
                            (open mol.db append mol_db)
                            (writeterm mol_db ?name]
                                          [write mol_db * *)
                                          (writeterm mol_db ?smstr)
                                          (writeln mol_db * * ?atlist * * ?bdlist)
                            (println *     Saved * ?name * to the Molecule Knowledge base... *)
                            (close mol_db)
              )

; (load_mol) - Loads information about a molecule from a MOL file created
;             by any MDL program into the molecule knowledge base.  User is prompted for
;             the MOL filename, and *ld_mol* does the loading.
              ((load_mol) if
                            (print *     File spec (including extension): *)
                            (getline stdin ?f)
                            (ld_mol ?f ?)
              )

; (ld_mol ?mol ?name) - Loads information about a molecule from the MOL
;             file ?mol created by any MDL program into the molecule knowledge base.
;             Returns the name of the molecule in ?name.  It first gets the
;             molecule name and checks to see if it already exists.
              ((ld_mol ?mol ?name) if
                            (open ?mol read molfile)
                            (getline molfile ?a)                    ; Molecule name
                            (molname ?a ?tname)
                            (close molfile)             ; Close Mol file in case already exists
                            (cut)
                            (check_molname ?tname)
                            (open ?mol read molfile)
                            (getline molfile ?b)                    ; Molecule name
                            (molname ?b ?name)
                            (getline molfile ?)                     ;   Don't want CAS #, etc
                            (getline molfile ?)                     ;   Blank line
                            (sread molfile ?at)                     ; Number of atoms
                            (sread molfile ?bd)                     ; Number of bonds
                            (getline molfile ?)                     ;   Read rest of line
;                           (println *     File name: {* ?mol *},*)
;                             (println *       Molecule name: {* ?name *}*)
;                           (println *       Atoms: * ?at *  Bonds: * ?bd);
                            (atoml ?at ?at () ?atlist)              ; Construct atom list
                            (bondl ?bd () ?bdlist)                  ; Construct bond list
                            (assert ((molecule ?name as ?atlist ?bdlist)) )         ; assert data
                            (close molfile)                                         ; Close Mol file
                            (cut)
              )

; (check_molname ?name) - Check to see if the molecule about to be input
;             already exists within the runtime molecule knowledge base.
;             Case 1: Molecule ?name doesn't exist in the runtime DB.
;             Case 2: Molecule ?name already exists, so ask the user if they would
;                           like to replace the currently existing molecule with the one
;                           about to be input.  If so, 'Y' or 'y' must be typed for the
;                           existing molecule to be retracted.  Cut to prevent backtracking.
;  NOTE - Molecules are only saved to the stored DB once the Smiles String
;                           is determined (see check_ss).
;               - If the user doesn't want it replaced, case 2 will fail.  If the
;                           system comes back again and asks the question, than duplicate
;                           entries for that molecule exist WHICH MUST BE CORRECTED.
              ((check_molname ?name) if
                            (not (molecule ?name ? ? ?)) (cut)                      )
              ((check_molname ?name) if
                            (molecule ?name ?ss ? ?)
                            (print *\nNote that the Molecule * ?name * with Smiles string\n*)
                              (print *     [* ?ss *] already exists in the runtime Molecule *)
                              (print *Knowledge base.\n      Would you like to replace this *)
                              (print *existing Molecule?\n      (Y or N, N by default) -> *)
                            (read ?Char)
                            (:= ?ane [strcase ?Char upper])
                            (= *Y* ?ane)
                            (println *          Replacing Molecule * ?name *.....*)
                            (retract ((molecule ?name ?ss ? ?)) )
                            (cut)
              )

; (check_molfrag ?name ?ss) - Check to see if fragment information
;             already exists for the molecule ?name within the runtime molecule
;             fragment knowledge base 'molfrag'.
;             Case 1: Fragment info for molecule ?name doesn't exist in the runtime DB.
;             Case 2: Fragment info for molecule ?name already exists, so ask the
;                           user if they would like to replace the currently existing fragment
;                           information.  If so, 'Y' or 'y' must be typed for the
;                           existing molecule to be retracted.  Cut to prevent backtracking.
;  NOTE - If the user doesn't want it replaced, case 2 will fail.  If the
;                           system comes back again and asks the question, than duplicate
;                           entries for that molecule exist WHICH MUST BE CORRECTED.
              ((check_molfrag ?name ?ss) if
                            (not (molfrag ?ss ? ? ? ?))                 )               ;don't know what frag #8's
are yet
              ((check_molfrag ?name ?ss) if
                            (molfrag ?ss ?ss1 ?ss2 ?ss3 ?ss4)
                            (println *\nNote that the Molecule * ?name * with Smiles*)
                              (println *      string [* ?ss *] and Smiles string list*)
                              (println *                [* ?ss1 * * ?ss2 * * ?ss3 * * ?ss4 *]*)
                              (print *      already exists in the runtime Molecule fragment *)
                              (print *Knowledge base.\n      [Note that if two chiral centers are *)
                              (print *present, you should proceed.]\n      Would you like to *)
                              (print *proceed ? (Y or N, N by default) -> *)
                            (cut)
                            (read ?Char)
                            (:= ?ane [strcase ?Char upper])
                            (= *Y* ?ane)
                            (println *          Replacing Molecule * ?name * fragment info....*)
                            (retract ((molfrag ?ss ? ? ? ?))            )
                            (cut)
              )

; (fragment_mol ?c_anum ?atlist ?bdlist ?ans) - Split the molecule into
;             four fragments around the chiral carbon atom number ?c_anum.  The
;             molecules atom list ?atlist and bond list ?bdlist are provided.  ?ans
;             is a list of fragment lists, each of which contains the chiral carbon
;             number as its first member.
;  NOTE - If no bond is found for the fourth fragment, it ASSUMES that there is
;                           is a Hydrogen attached.
              ((fragment_mol ?c_anum ?atlist ?bdlist ?ans) if
                            (is_bond ?bdlist ?c_anum ?A1)                                       ;Find 1st bond w/chrl
                            (listcat (?c_anum) (?A1) ?list1)                                    ; add to fgmt list
                            (frag_let ?aElist ?bdlist ?A1 nil ?list1 ?ans1)                     ; get its fgmt list
;                                       (println *      The first fragment is * ?ans1 *.*)
                            (is_bond ?bdlist ?c_anum ?A2)                                       ;Find 2nd bond w/chrl
                            (not (= ?A2 ?A1))                                                   ; Not same as 1st
                            (listcat (?c_anum) (?A2) ?list2)                                    ; add to fgmt list
                            (frag_let ?aElist ?bdlist ?A2 nil ?list2 ?ans2)                     ; get its fgmt list
;                                       (println *      The second fragment is * ?ans2 *.*)
                            (is_bond ?bdlist ?c_anum ?A3)                                       ;Find 3rd bond w/chrl
                            (not (= ?A3 ?A1))           (not (= ?A3 ?A2))                       ; Not same as 1st or 2nd
                            (listcat (?c_anum) (?A3) ?list3)                                    ; add to fgmt list
                            (frag_let ?aElist ?bdlist ?A3 nil ?list3 ?ans3)                     ; get its fgmt list
;                                       (println *      The Third fragment is * ?ans3 *.*)
                            (cut)
                            (or                                                                 ;The 4th bond either
                               (                                                                ;a) exists, so use it or
                                 (is_bond ?bdlist ?c_anum ?A4)
                                 (not (= ?A4 ?A1))          (not (= ?A4 ?A2))                   (not (= ?A4 ?A3))
                                 (listcat (?c_anum) (?A4) ?list4)
                                 (frag_let ?aElist ?bdlist ?A4 nil ?list4 ?ans4)
;                                          (println *      The fourth fragment is * ?ans4 *.*)
                                 (order_fraglst ?ans1 ?ans2 ?ans3 ?ans4 ?ans)
                               )
                          ( (:= ?ans ?ansB)                                                     ;b) it's a Hydrogen
                                 (listcat (?A1) (?A2) ?listA)                                   ;Build list of known bonds
                                 (listcat ?listA (?A3) ?listB)                                  ; to chiral atom
                                 (attached_H ?atlist ?bdlist ?c_anum ?listB)
                                 (order_fraglst ?ans1 ?ans2 ?ans3 () ?ane)
                                 (println *      The fourth fragment is a Hydrogen....")
                               )
                                                                ; end OR
                            )                                   ;*** END fragment_mol ?c_anum ?atlist ?bdlist ?ans

; (frag_let ?atlist ?bdlist ?c_anum ?elist ?flist ?ans) - Construct a
;             list of all atoms connected to the current atom number ?c_anum
;             that are not already contained in the fragment list ?flist.
;             ?flist is built as bonds with new atoms are discovered.  When
;             first called,  ?c_anum will be one of the atoms around a Chiral
;             atom, and ?flist will contain the chiral atom number and ?c_anum.
;             The predicate uses the atom and bond lists ?atlist and ?bdlist,
;             and case 4 returns the fragment list in ?ans, ?ans will be a
;             list of single atom numbers, with the chiral carbon being the
;             first member of that list.  The search list ?elist contains a
;             list of atoms that have not been thoroughly investigated for
;             bonds to other atoms.
;             Case 1: A New bond has been found with current atom number ?c_anum,
;                           and there are bonds yet to be discovered.  A dept first search
;                           to find new atoms is used.  Newly discovered atoms bonded with
;                           ?c_anum are added to ?flist, ?c_anum is added to ?elist (doing
;                           a depth first search, so it needs to be looked at further),
;                           and recursively call itself with the newly discovered atom ?A2
;                           as ?c_anum.
;             Case 2: Since there are no more new bonds with ?c_anum, take the
;                           first member off of the search list and recursively call itself
;                           with this atom number as ?c_anum.
;             Case 4: No more bonds on search list, so return the fragment list
;                                       ?flist in ?ans.
; NOTE - Rings need not be tested for, since all atoms of the ring will
;             be found with cases 1&2.  Simply recognizing the closure of a ring
;             structure does not add any needed information.
;             ((frag_let ?atlist ?bdlist ?c_anum ?elist ?flist ?ans) if
;                           (print *\n    Case 1: ?c_anum=* ?c_anum *, ?elist=* ?elist)
```

```
;                       (println ", ?flist=" ?flist ".")
                (is_bond ?bdlist ?c_anum ?A2)           ;is atom bonded to ?c_anum that
                (not (member ?A2 ?flist))               ; isn't member of current ?flist?
;               (println "    Case 1: found other atom " ?A2 ", not a member.")
                (listcat ?flist (?A2) ?flist2)          ;if so, add it to new flist and
                (listcat ?alist (?c_anum) ?alist2)      ; store c_anum on search list
;               (println "    Case 1: Search1 " ?alist2 ", fraction1 " ?flist2 ".")
                (frag_lst ?atlist ?bdlist ?A2 ?alist2 ?flist2 ?ans)
)                                                       ; go on with newly found atom
((frag_lst ?atlist ?bdlist ?c_anum (?fst . ?rst) ?flist ?ans) if
;               (print "\n    Case 2: drop ?c_anum=" ?c_anum ", ?fst=" ?fst)
;               (println ", ?rst=" ?rst ", ?flist=" ?flist ".")
                (frag_lst ?atlist ?bdlist ?fst ?rst ?flist ?ans)
)
((frag_lst ? ? ? nil ?flist ?flist))    ;no more bonds on alist, so stop

; (order_fraglst ?ans1 ?ans2 ?ans3 ?ans4 ?ans) - Sort the 4 fragment lists
;            ?ans1 - ?ans4 according to the number of atoms contained in each.
;            The idea is to have ?ans1 to be the largest number, followed by ?ans2,
;            ?ans 3 and finally ?ans4.  It in effect, this predicate performs a
;            bubble sort on all of the values.  Case 4 returns the answer in ?ans.
;            Case 1 - If the size of ?ans1 is less than the size of ?ans2, switch
;                     them and recursively continue.  If this isn't true, than the two
;                     are already in the correct order so go on to the next case.
;            Case 2 - If the size of ?ans2 is less than the size of ?ans3, switch
;                     them and recursively continue.  If this isn't true, than the two
;                     are already in the correct order so go on to the next case.
;            Case 3 - If the size of ?ans3 is less than the size of ?ans4, switch
;                     them and recursively continue.  If this isn't true, than the two
;                     are already in the correct order so go on to the next case.
;            Case 4 - Final case....All the values are in order by decreasing size.
;                     Combine all of them and return the list in ?ans.
((order_fraglst ?ans1 ?ans2 ?ans3 ?ans4 ?ans) if        ;?ans1&2 out of order
                (listsiz ?ans1 ?size1) (listsiz ?ans2 ?size2)
                (< ?size1 ?size2)
                (order_fraglst ?ans2 ?ans1 ?ans3 ?ans4 ?ans)) ;Switch ?ans1&2, redo
((order_fraglst ?ans1 ?ans2 ?ans3 ?ans4 ?ans) if        ;?ans2&3 out of order
                (listsiz ?ans2 ?size2) (listsiz ?ans3 ?size3)
                (< ?size2 ?size3)
                (order_fraglst ?ans1 ?ans3 ?ans2 ?ans4 ?ans)) ;Switch ?ans2&3, redo
((order_fraglst ?ans1 ?ans2 ?ans3 ?ans4 ?ans) if        ;?ans3&4 out of order
                (listsiz ?ans3 ?size3) (listsiz ?ans4 ?size4)
                (< ?size3 ?size4)
                (order_fraglst ?ans1 ?ans2 ?ans4 ?ans3 ?ans)) ;Switch ?ans3&4, redo
((order_fraglst ?ans1 ?ans2 ?ans3 ?ans4 ?ans) if
                (listcat (?ans1) (?ans2) ?ansA)                 ;build answer list
                (listcat ?ansA (?ans3) ?ansB) (listcat ?ansB (?ans4) ?ans))

; (save_frag_mol ?name ?atlist ?bdlist ?flist ?filename ?filecnt) - Save
;            the smiles string file for each of the fragment lists sent in ?flist.
;            Assumes that ?flist is sorted w/rt size, and any hydrogens will be
;            last in the list.
;            Case 1: Fragment list empty, so stop.
;            Case 2: Fragment list has H as last member, so stop.
;            Case 3: Split off the first fragment list (removing the chiral carbon
;                    number ?cnum), find the new atom and bond lists, get the correct
;                    file name and call smiles_str to save the smiles string.
;  NOTE - Each of the fragment lists removes the first member, ie the chiral
;         carbon number, from the fragment list before proceeding.
                ((save_frag_mol ? ? ? nil ? ?))                 ;When last fragment isn't a H
                ((save_frag_mol ? ? ? (nil.nil) ? ?))   ;When last fragment is a H
                ((save_frag_mol ?name ?atlist ?bdlist ((?cnum.?flst).?rst) ?fn ?fcnt) if
                (renum_fl ?flst ?flist ?)
;               (println "  The renumbered fragment list is\n     [" ?flist "].")
                (new_atlist ?atlist ?atl ?flist)
;               (println "     The renumbered atom list is\n       (" ?atl ").")
                (new_bdlist ?bdlist ?bdl ?flist)
;               (println "     The renumbered bond list is\n       (" ?bdl ").")
                (:= ?fname [strcat ?fn [numtostr ?fcnt 0]])                     ;Make file name
                (smiles_str ?name ? ?atl ?bdl ?fname)                           ;Save as file
;               (println "     The file saved as [" ?fname "].")
                (:= ?fcnt1 [+ ?fcnt 1])
                (save_frag_mol ?name ?atlist ?bdlist ?rst ?fn ?fcnt1)
)

; (renum_fl ?flist ?rflist ?rfnum) - The renumbered fragment list ?rflist
;            is generated from the fragment list ?flist.  Breaks down the
;            fragment list recursively, and after the end is reached, assigns
;            each member of the fragment a new number and construct the renumbered
;            fragment list as the predicate recurses out.  ?rfnum keeps count of
;            the new fragment numbers.
;            Case 1 - when end of fragment list is reached, start the renumbered
;                     fragment list and initialize the new fragment numbers.
;            Case 2 - renumber the current member of the fragment list and
;                     concatinate it to build the renumbered fragment list
; NOTE - ?flist is a list of single numbers, whereas ?rflist is a list
;        of pairs of numbers.
                ((renum_fl nil nil 1))
                ((renum_fl (?fst.?rst) ?rflist ?rfnum) if
                (renum_fl ?rst ?rflist1 ?rfnum1)                        ;break down fragment list
                (listcat (?fst) (?rfnum1) ?rlist)                       ;renum ?fst to num ?rfnum1
                (listcat (?rlist) ?rflist1 ?rflist)     ;add this set to new flist
                (:= ?rfnum (+ ?rfnum1 1))               ;increment new fragment num
)
; (new_atlist ?atlist ?atl ?flist) - ?atlist contains the complete atom
;            list for the molecule, ?flist is the fragment list currently being
;            used to construct a new atom list ?atl.  This predicate recursively
;            breaks down ?atlist until the end is reached, than inspects each
;            atom as it recurses out, adding them to ?atl if the atom appears
;            in ?flist.  ?flist is a list of pairs of numbers.
;            Case 1: No more atoms in the atom list, so initialize the new
;                    atom list ?atl and begin recursing out.
;            Case 2: predicate "conv_atom has determined that the atom at head
;                    of the original atom list is a member of the fragment list, so
;                    it must be copied to the new atom list ?atl.
;            Case 3: Atom at head of atom list is NOT a member of the fragment list
;                    so it is dropped.
((new_atlist nil nil ?))
((new_atlist ((atom ?anum ?elm ?num).?rst) ?atl2 ?flist) if
                (conv_atom ?anum ?rnum ?flist)
;               (println "     Atom number " ?anum " renumbered as " ?rnum ".")
                (new_atlist ?rst ?atl ?flist)
                (listcat (atom) (?rnum) ?liste)         ; build atom entry for
                (listcat ?liste (?elm) ?listb)          ;   molecular knowledge base
                (listcat ?listb (?num) ?listc)
                (listcat ?atl (?listc) ?atl2)
)
((new_atlist (?fst.?rst) ?atl ?flist) if
;               (println "      The atom " ?fst " isn't a member.")
                (new_atlist ?rst ?atl ?flist) )

; (new_bdlist ?bdlist ?bdl ?flist) - ?bdlist contains the complete bond
;            list for the molecule, ?flist is the fragment list currently being
;            used to construct a new bond list ?bdl.  This predicate recursively
;            breaks down ?bdlist until the end is reached, than inspects each
;            bond as it recurses out, adding them to ?bdl if both atom in the
;            bond appear in ?flist.  ?flist is a list of pairs of numbers.
;            Case 1: No more bonds in the bond list, so initialize the new bond
;                    list ?bdl and begin recursing out.
;            Case 2: predicate "conv_atom" has determined that both atoms in the
;                    current bond are members of the fragment list, so this bond must
;                    be copied to the new bond list ?bdl.
;            Case 3: the current bond is NOT a member of the fragment list, so drop it.
((new_bdlist nil nil ?))
((new_bdlist ((bond ?elm1 ?elm2 ?bnd).?rst) ?bdl2 ?flist) if
                (conv_atom ?elm1 ?relm1 ?flist)
                (conv_atom ?elm2 ?relm2 ?flist)
;               (printf "     (bond " ?elm1 " " ?elm2 " " ?bnd ") renumbered ")
;               (println "(bond " ?relm1 " " ?relm2 " " ?bnd ")")
                (new_bdlist ?rst ?bdl ?flist)
                (listcat (bond) (?relm1) ?liste)        ; build bond entry for
                (listcat ?liste (?relm2) ?listb)        ;   molecular knowledge base
                (listcat ?listb (?bnd) ?listc)
                (listcat ?bdl (?listc) ?bdl2)
)
((new_bdlist (?fst.?rst) ?bdl ?flist) if
;               (println "      The bond " ?fst " isn't a member.")
                (new_bdlist ?rst ?bdl ?flist)           )

; (atoml ?numat ?atcnt ?atl ?atlist) - Constructs the atom list for a
;            molecule from information in an MDL Mol file.  Assumes there are
;            ?numat number of atoms in the molecule, EXCLUDING hydrogen atoms
;            (as stipulated by MDL's Mol file).  Is currently processing atom
;            number ?atcnt.  Uses ?atl as a list used to build up the atom
;            list, and returns it in ?atlist.
;            Case 1: when there are no more atoms to process, return the atom
;                    list in ?atlist.
;            Case 2: While there are atoms to process, get the information on
;                    the current atom and recursively call itself to process the
;                    rest of the atom list
((atoml ?numat 0 ?atlist ?atlist))
((atoml ?numat ?atcnt ?atl ?atlist) if
                (mread molfile ?)                                       ;  Don't want X coordinate
                (mread molfile ?)                                       ;  Don't want Y coordinate
                (mread molfile ?)                                       ;  Don't want Z coordinate
                (mread molfile ?element)    ; read element symbol
                (mread molfile ?)                                       ;  Don't want mass difference
                (mread molfile ?)                                       ;  Don't want charge code
                (mread molfile ?chiral)                         ; read stereo parity indicator
                (getline molfile ?)                             ;  Don't want rest of info on line
                (:= ?num [- ?atcnt 1])                          ; reduce atom count by one
                (:= ?atnum [- ?numat ?num])                             ; calculate atom number
                (listcat (atom) (?atnum) ?liste)        ; build atom entry for
                (listcat ?liste (?element) ?listb)      ;   molecular knowledge base
                (listcat ?listb (?chiral) ?listc)
                (listcat ?atl (?listc) ?listd)
;               (println "     atom : " ?listd)          ; print out current atom info
                (atoml ?numat ?num ?listd ?atlist)      ; recursively process rest of list
)

; (bondl ?bdnum ?bdl ?bdlist) - Constructs the bond list for a molecule
;            from information in an MDL Mol file.  Assumes there are ?bdnum
;            number of bonds in the molecule, EXCLUDING hydrogen atom bonds
```

```
;             (as stipulated by MDL's Mol file).  Uses ?bdl as a list used to
;             build up the bond list, and returns the list in ?bdliet.
;             Case 1: when there are no more atoms to process, return the bond
;                     list in ?bdliet.
;             Case 2: While there are bonds to process, get the information on
;                     the current bond and recursively call itself to process the
;                     rest of the bond list
              ((bondl 0 ?bdliet ?bdliet))
              ((bondl ?bdcnt ?bdl ?bdliet) if
                      (eread molfile ?bd1)                    ; read first atom in bond
                      (eread molfile ?bd2)                    ; read second atom in bond
                      (eread molfile ?bdtype)                 ; read bond type
                      (getline molfile ?)                     ;  Don't want rest of info
                      (:= ?num (- ?bdcnt 1))                  ; reduce bond count by one
                      (listcat (bond) (?bd1) ?liata)          ; build bond entry for
                      (listcat ?liata (?bd2) ?liatb)          ;   molecular knowledge base
                      (listcat ?liatb (?bdtype) ?liatc)
                      (listcat ?bdl (?liatc) ?liatd)
;                     (println * bond : * ?liatd)             ; print out current bond info
                      (bondl ?num ?liatd ?bdliet)             ; recursively process rest of list
              )

; (molname ?a ?b) - takes the first string passed to it and removes all
;             trailing spaces, returning the original string less trailing spaces.
              ((molname ?a ?b) if
                      (:= ?pos (strpos ?a "     " 0))
                      (> ?pos 0)
                      (:= ?b (strsub ?a 0 ?pos))              )
              ((molname ?a ?a) if (cut) )

;   ***END OF assert
)


;                             **** File CHRM.HC ****                  February 26, 1993
(assert

; (valence ?atom ?valence ?mul_factor) - Facts for the Index of Hydrogen
;             Deficiency example.  Consist of the fact name "valence", the atomic
;             symbol, the valence, and the multiplication factor for the atom.  Can
;             handle atoms with multiple valences if predicates that use it also
;             implement a "fail".
              ((valence H 1 -1))
              ((valence Cl 1 -1))
              ((valence Br 1 -1))
              ((valence O 2 0))
              ((valence N 3 1))
              ((valence C 4 2))

; (Index_H_D ?liat) - Calculates the Index of Hydrogen Deficiency (IHD) for
;             the molecule whose molecular formula is passed in as ?liat.  The IHD
;             is defined as IHD = [ (2*Tetravalent + 2 + Trivalent) - Monovalent] /2
              ((Index_H_D ?liat) if
                      (count_num ?liat ?count)
                      (:= ?ans [/ [+ ?count 2] 2])
                      (print "      The answer is " ?ans ".")
              )

; (count_num ?liat ?count) - Recursively calls itself to calculate the IHD
;             for individual atoms in the molecular formula ?liat.
;             Case 1: Count adds up the value for individual atoms as it recurses out.
;             Case 2: Initializes the IHD count
              ((count_num ((?atom ?number) . ?rest) ?cnt) if
                      (valence ?atom ? ?mult_fact)
                      (:= ?cnt2 [* ?number ?mult_fact])
                      (count_num ?rest ?cnt3)
                      (:= ?cnt [+ ?cnt2 ?cnt3)]
              )
              ((count_num nil 0))

; (num_rings ?name) - Calculates the number of rings contained in the molecule
;             ?name.  ?name must be found in the molecule KB.  Uses "fail" to handle
;             atoms that might have multiple valences.
              ((num_rings ?name) if
                      (molecule ?name ?as ?atliat ?bdliat)
                      (println "Looking at molecule " ?name "....")
                      (cnt_rings ?atliat ?bdliat ?num)
                      (println "    Molecule " ?name " has " ?num " ring(s).")
                      (fail)
              )

; (cnt_rings ?atliat ?bdliat ?num) - Counts the number of rings found within
;             the molecule that has the atom list ?atliat and bond list ?bdliat.  It
;             first determines the number of different atoms that are present, then
;             the number of bonds.  Lastly, it uses all of these values to calculate
;             first the IHD of the molecule, then subtracts the number of double
;             bonds and twice the number of tripple bonds to arrive at the number of
;             rings.
              ((cnt_rings ?atliat ?bdliat ?num) if
                      (cnt_atoms ?atliat "H" ?Hnum)
                      (cnt_atoms ?atliat "C" ?Cnum)
                      (cnt_atoms ?atliat "N" ?Nnum)
                      (cnt_atoms ?atliat "F" ?Fnum)
                      (cnt_atoms ?atliat "Cl" ?Clnum)
                      (cnt_atoms ?atliat "Br" ?Brnum)
                      (cnt_atoms ?atliat "I" ?Inum)
                      (cnt_bonds ?bdliat 2 ?dblnum)
                      (cnt_bonds ?bdliat 3 ?tplnum)
                      (:= ?numdbonds [+ ?dblnum [* 2 ?tplnum]])
                      (cnt_H "C" 4 ?atliat ?bdliat ?CnumH)
                      (cnt_H "N" 3 ?atliat ?bdliat ?NnumH)
                      (cnt_H "O" 2 ?atliat ?bdliat ?OnumH)
                      (:= ?halogens [+ ?Fnum [+ ?Clnum [+ ?Brnum ?Inum]]])
                      (:= ?numH [+ ?halogens [+ ?OnumH [+ ?CnumH ?NnumH]]])
                                          ; Could subtract ?Hnum from ?numH above
                      (:= ?num [- [/ [- [+ ?Nnum [+ 2 [* 2 ?Cnum]]] ?numH] 2] ?numdbonds])
                      (cut)
              )

; (cnt_H ?atsy ?nbonds ?atliat ?bdliat ?numH) - Counts the number of hydrogen
;             atoms that are left unspecified for each atom of type ?atsy within the
;             molecule with the atom list ?atliat and bond list ?bdliat.
;             Case 1: Base case, initializes the unspecified hydrogen count at zero.
;             Case 2: Found an atom of type ?atsy, so call cnt_atombonds to see how
;                     many single, double, and tripple bonds exist within the bond list
;                     for the atom.  Recursively call itself to process the rest of the
;                     list, counting them up as it recurses out.
;             Case 3: Current atom doesn't have any bonds, so go on to the next one.
              ((cnt_H ?atsy ?nbonds nil ?bdliat 0))
              ((cnt_H ?atsy ?nbonds ((atom ?atnum ?atsy ?) ?rst) ?bdliat ?numH) if
                      (cnt_atombonds ?atnum 1 ?bdliat ?Snum)
                      (cnt_atombonds ?atnum 2 ?bdliat ?Dnum)
                      (cnt_atombonds ?atnum 3 ?bdliat ?Tnum)
                      (cnt_H ?atsy ?nbonds ?rst ?bdliat ?numH2)
                      (:= ?numH [+ ?numH2 [- ?nbonds [+ ?Snum [+ ?Dnum ?Tnum]]]])
              )
              ((cnt_H ?atsy ?nbonds (?fst.?rst) ?bdliat ?numH) if
                      (cnt_H ?atsy ?nbonds ?rst ?bdliat ?numH))

; (cnt_atombonds ?atnum ?bdtype ?bdliat ?num) - Counts the number of bonds of
;             order ?bdtype for a particular atom number ?atnum within the atom list.
;             This task counts the number of times the atom number ?atnum appears in
;             the bond list ?bdliat associated with a bond of order ?bdtype.
;             Case 1: Base case, initializes the count at zero.
;             Case 2: Finds a bond of type ?bdtype with the atom number ?atnum as the
;                     first atom in the bond.
;             Case 3: Finds a bond of type ?bdtype with the atom number ?atnum as the
;                     second atom in the bond.
;             Case 4: Current bond in the bond list does not involve atom ?atnum, so
;                     go on to the next bond.
              ((cnt_atombonds ?atnum ?bdtype nil 0))
              ((cnt_atombonds ?atnum ?bdtype ((bond ?atnum ? ?bdtype).?rst) ?num) if
                      (cnt_atombonds ?atnum ?bdtype ?rst ?num2)
                      (:= ?num [+ ?num2 ?bdtype])                                    )
              ((cnt_atombonds ?atnum ?bdtype ((bond ? ?atnum ?bdtype).?rst) ?num) if
                      (cnt_atombonds ?atnum ?bdtype ?rst ?num2)
                      (:= ?num (+ ?num2 ?bdtype))                                    )
              ((cnt_atombonds ?atnum ?bdtype (?fst.?rst) ?num) if
                      (cnt_atombonds ?atnum ?bdtype ?rst ?num)                       )

; (cnt_atoms ?atliat ?atomicsymbol ?cnt) - Count the number of atoms within
;             the molecule with the atom list ?atliat of type ?atomicsymbol.  Return
;             this number in ?cnt.
;             Case 1: Initialize the number of atoms at zero.
;             Case 2: Finds an atom ?atsy, recursively calls itself to search the rest
;                     of the list, and adds one to the count as it recurses out.
;             Case 3: Didn't find an atom ?atsy at this member of the atom list, so go
;                     on to the next member of the atom list.
              ((cnt_atoms nil ? 0))
              ((cnt_atoms ((atom ? ?atsy ?).?rst) ?atsy ?cnt) if
                      (cnt_atoms ?rst ?atsy ?cnt2)
                      (:= ?cnt [+ 1 ?cnt2])                          )
              ((cnt_atoms (?fst.?rst) ?atsy ?cnt) if
                      (cnt_atoms ?rst ?atsy ?cnt)                    )

; (cnt_bonds ?bdliat ?bondorder ?numbonds) - Count the number of bonds within
;             the molecule with the bondliat ?bdliat that have a bond order of
;             ?bondorder.  Return the number in ?numbonds.
;             Case 1: Initializes the number of bounds at zero.
;             Case 2: Finds a bond of order ?bdor, recursively calls itself to search
;                     the rest of the list, and adds one to the count as it recurses out.
;             Case 3: Didn't find a bond of order ?bdor at this member of the bond list
;                     so go on to the next member of the bond list.
              ((cnt_bonds nil ? 0))
              ((cnt_bonds ((bond ? ? ?bdor).?rst) ?bdor ?cnt) if
                      (cnt_bonds ?rst ?bdor ?cnt2)
                      (:= ?cnt [+ 1 ?cnt2])                          )
              ((cnt_bonds (?fst.?rst) ?bdor ?cnt) if
                      (cnt_bonds ?rst ?bdor ?cnt)                    )

; (find_chiral ?atliat ?cnum) - finds the FIRST chiral atom in the atom
;             list ?atliat.  Returns the chiral atom's number in ?cnum.
; NOTE - Assumes that the chiral atom will be a CARBON, and that HAYDN will
;             assign it as type 1(clockwise), 2(counterclockwise), or 3(either).
```

```
                ((find_chiral nil 0) if
                        (println *          ***** No Chiral atom found!*)
                        (cut)   )
                ((find_chiral ((atom ?Anum C 1) . ?) ?Anum))
                ((find_chiral ((atom ?Anum C 2) . ?) ?Anum))
                ((find_chiral ((atom ?Anum C 3) . ?) ?Anum))
                ((find_chiral ((atom ? ? ?) . ?ret) ?cnum) if
                        (find_chiral ?ret ?cnum)    )

; (findsb ?bdlist ?A1 ?A2) - Finds single bonds in the bond list ?bdlist,
;    and returns the numbers ?A1 & ?A2 of atoms on either end of the bond.
;    Case 1 & 2: is a single bond at the head of the bond list, return
;                Values for the two atom numbers.
;    Case 3: Head is not a single bond, recursively call function with
;                the rest of the list.
;  NOTE - an atom number can be supplied to search for specific double
;    bonds (see case 1 and 2).  If an atom number is NOT SUPPLIED,
;    that section of code MUST CONTAIN a statement such as "(< ?A1 ?A2)"
;    to prevent both Case 1 and 2 below from being satisified.
;    Refer to routine "has C=C" for an example
                ((findsb ((bond ?A1 ?A2 1) . ?ret) ?A1 ?A2) if
;                       (println *      Found s(bond * ?A1 * * ?A2 * 1).*)
                        )
                ((findsb ((bond ?A2 ?A1 1) . ?ret) ?A1 ?A2) if
;                       (println *      Found sr(bond * ?A1 * * ?A2 * 1).*)
                        )
                ((findsb (?1st . ?ret) ?A1 ?A2) if
;                       (println *      Passed s* ?1st *.*)
                        (findsb ?ret ?A1 ?A2)    )

; (ofindsb ?bdlist ?A1 ?A2 ?A1) - Finds single bonds with atom ?A1 in the
;    bond list ?bdlist other than atom ?A2.  Typically the atom number ?A1
;    and ?A2 are sent as a parameters.  This function is used after "findsb"
;    to find a second single bond from the atom ?A1 to another atom ?A3
;    where atom ?A2 is not ?A3.  If it exists, the function returns with
;    the other single bond between atoms ?A1 and ?A3.
                ((ofindsb ?bdlist ?A1 ?A2 ?A1) if
                        (findsb ?bdlist ?A1 ?A3)
;                       (println *  Second bond at * ?A1 * * ?A3 * found.  Name as first?*)
                        (not (= ?A2 ?A3))       ; Make sure it's not the same bond
;                       (println *  No, found a second single bond.*)
                )

; (finddb ?bdlist ?A1 ?A2) - Finds double bonds in the bond list ?bdlist,
;    and returns the numbers ?A1 & ?A2 of atoms on either end of the bond.
;    Case 1 & 2: is a double bond at the head of the bond list, return
;                values for the two atom numbers.
;    Case 3: Head is not a double bond, recursively call function with
;                the rest of the list.
;  NOTE - an atom number can be supplied to search for specific double
;    bonds (see case 1 and 2).  If an atom number is NOT SUPPLIED,
;    that section of code MUST CONTAIN a statement such as "(< ?A1 ?A2)"
;    to prevent both Case 1 and 2 below from being satisified.
;    Refer to routine "has C=C" for an example
                ((finddb ((bond ?A1 ?A2 2) . ?ret) ?A1 ?A2) if
;                       (println *      Found d(bond * ?A1 * * ?A2 * 2).*)
                        )
                ((finddb ((bond ?A2 ?A1 2) . ?ret) ?A1 ?A2) if
;                       (println *      Found dr(bond * ?A1 * * ?A2 * 2).*)
                        )
                ((finddb (?1st . ?ret) ?A1 ?A2) if
;                       (println *      Passed d* ?1st *.*)
                        (finddb ?ret ?A1 ?A2)    )

; (findtb ?bdlist ?A1 ?A2) - Finds tripple bonds in the bond list ?bdlist,
;               and returns the number ?A1 of the atoms on either end of the bond.
;               Case 1 & 2: is a tripple bond at the head of the bond list, return
;                           values for the two atom numbers.
;               Case 3: Head is not a tripple bond, recursively call function with
;                           the rest of the list.
;  NOTE - an atom number can be supplied to search for specific tripple
;               bonds (see case 1 and 2).  If an atom number is NOT SUPPLIED,
;               that section of code MUST CONTAIN a statement such as "(< ?A1 ?A2)"
;               to prevent both Case 1 and 2 below from being satisified.
;               Refer to routine "has C=N" for an example.
                ((findtb ((bond ?A1 ?A2 3) . ?ret) ?A1 ?A2) if
;                       (println *      Found t(bond * ?A1 * * ?A2 * 2).*)
                        )
                ((findtb ((bond ?A2 ?A1 3) . ?ret) ?A1 ?A2) if
;                       (println *      Found tr(bond * ?A1 * * ?A2 * 2).*)
                        )
                ((findtb (?1st . ?ret) ?A1 ?A2) if
;                       (println *      Passed t* ?1st *.*)
                        (findtb ?ret ?A1 ?A2)       )

; (is_bond ?bdlist ?A1 ?A2) - Determine if there are any bonds in ?bdlist
;               between atoms ?A1 and ?A2.  If neither atom is specified, the first
;               bond in ?bdlist will be unified.  Specifying both atoms will result
;               in determining if a bond exists between them.
;               Case 1 : Check if bond ?A1-?A2 matches the first bond in ?bdlist.
;               Case 2 : Check for bond ?A2-?A1, ie reverse listing in ?bdlist.
;               Case 3 : If the first bond in ?bdlist doesn't match, remove it
;                           and recursively check the remaining bonds in ?bdlist.
                ((is_bond ((bond ?A1 ?A2 ? ) . ?ret) ?A1 ?A2) if
;                       (println *      Case 1 Bond : ?A1 = * ?A1 * ?A2 = * ?A2)
                )
                ((is_bond ((bond ?A2 ?A1 ? ) . ?ret) ?A1 ?A2) if
;                       (println *      Case 2 Bond : ?A1 = * ?A1 * ?A2 = * ?A2)
                )
                ((is_bond ((bond ? ? ? ) . ?ret) ?A1 ?A2) if
;                       (println *      Searching for another bond ...*)
                        (is_bond ?ret ?A1 ?A2)
                )

; (bond_type ?bdlist ?A1 ?A2 ?type) - Determine if a bond between atom
;               ?A1 and atom ?A2 of type ?type exists in the bond list ?bdlist.
;               If no bond type is given, the type will be returned.
;               Case 1: Check for bond ?A1-?A2
;               Case 2: Check for bond ?A2-?A1
                ((bond_type ?bdlist ?A1 ?A2 ?type) if
                        (member (bond ?A1 ?A2 ?type) ?bdlist)
                        (cut)
                )
                ((bond_type ?bdlist ?A1 ?A2 ?type) if
                        (member (bond ?A2 ?A1 ?type) ?bdlist)
                )

; (is_atom ?atlist ?atsymb ?Anum) - Looks for the atomic symbol ?atsymb
;    in the atom list ?atlist, and if it is found returns the atom
;    number in ?Anum.  Can also be used to check if atom ?Anum is the
;               element ?atsymb.
                ((is_atom ((atom ?Anum ?atsymb ?) . ?ret) ?atsymb ?Anum))
                ((is_atom (?1st . ?ret) ?atsymb ?Anum) if
;                       (print *   Passing atom list, no match of * ?1st * with * ?atsymb)
;                       (println *.*)
                        (is_atom ?ret ?atsymb ?Anum)    )

; (attached_H ?atlist ?bdlist ?A1 ?A2) - Succeeds if there is a hydrogen
;               attached to atom ?A1.  IT WILL FAIL if it finds an atom bonded to ?A1
;               that is either not a Hydrogen or not found in list ?A2.  ?A2
;               contains a list of all the known atoms known and expected to
;               be bonded to atom ?A1.  YOU MUST SEND a list of all single bonds
;               with ?A1 that are expected in the list ?A2.  Double bonds need not
;               be specified (see end of note below).
;               Case 1: Succeeds if you CAN NOT find another single bond NOT already
;                       in the list of known atoms.  There are no other single bonds to
;                       atom ?A1.
;               Case 2: Succeeds only if the single bond found that is not in the list
;                       of known atoms is a Hydrogen.
;  NOTE - If ?A1 is 'O', succeeding means it must be an OH.
;               If ?A1 is 'C', has no db's or other sb's,
;                       and ?A2 has 1 member, than succeeding means RCH3.
;                       As above with ?A2 having 2 members, than R2CH2.
;                       As above with ?A2 having 3 members, than R3CH.
;               If ?A1 is 'C', has a db, has no other sb's,
;                       and ?A2 has 1 member(not the db), than succeeding means RC=OH.
;               As above with ?A2 having 2 members, THAN DON'T USE this predicate
;                       (although it would succeed since it doesn't check for more
;                       than 4 bonds with a carbon atom).
                ((attached_H ? ?bdlist ?A1 ?A2) if
;                       (println *          atc_H C1: no more sb w/ * ?A1 * not in [* ?A2 *].*)
                        (not        (and (
                                    (findsb ?bdlist ?A1 ?A3)
                                    (not (member ?A3 ?A2))
                                                )                   )
                                    ;Once you find an H at ?A1, don't want
                        (cut)               )                       ; to rpt other H's at ?A1, so CUT
                ((attached_H ?atlist ?bdlist ?A1 ?A2) if
;                       (println *          atc_H C2: have bond w/* ?A1 * not in [* ?A3 *]?*)
                        (findsb ?bdlist ?A1 ?A3)                    ;If a single bond is found
                        (not (member ?A3 ?A2))                      ; that is not a member of ?A2, then
;                       (println *      found [* ?A1 *-* ?A3 *] C2: is it H?*)
                        (is_atom ?atlist "H" ?A3)  ; Make sure it is a hydrogen
                        (cut)               )       ;Don't want to rpt other H's at ?A1, so CUT

;   ***END OF assert
)

;                               **** File PG.HC ****                        February 26, 1993
(assert

                ((CdC ?name) if
                        (has_CdC ?name ?A1 ?A2)
                        (println "*    Found C=C in * ?name * between * ?A1 * and * ?A2 *.*)
;                       (fail)
                )

                ((CtC ?name) if
                        (has_CtC ?name ?A1 ?A2)
                        (println "*    Found -CC- in * ?name * between * ?A1 * and * ?A2 *.*)
;                       (fail)
                )
```

```
((CtN ?name) if
                (has_CtN ?name ?A1 ?A2)
                (println "    Found -CN in " ?name " at " ?A1 " [" ?A2 "].")
;               (fail)
 )

((Oath ?name) if
                (has_Oath ?name ?A1 ?A2)
                (print "    Found C-O-C bond in " ?name " at Oxygen " ?A1 " " ?A2)
                (println ?A2 ".")
;               (fail)
 )

((OaH ?name) if
                (has_OaH ?name ?A1 ?A2)
                (println "    Found -OH bond in " ?name " at atom " ?A1 ".")
;               (fail)
 )

((CdO ?name) if
                (has_CdO ?name ?A1 ?A2)
                (println "    Found C=O bond in " ?name " at atom " ?A1 ".")
;               (fail)
 )

((CHdO ?name) if
                (has_CHdO ?name ?A1 ?A2)
                (println "    Found Aldehyde (CH=O) in " ?name " at atom " ?A1 ".")
;               (fail)
 )

((X2CdO ?name) if
                (has_X2CdO ?name ?A1 ?A2)
                (println "    Found Keytone (X2C=O) in " ?name " at atom " ?A1 ".")
;               (fail)
 )

((COOH ?name) if
                (has_COOH ?name ?A1 ?A2)
                (print "    Found Carboxylic Acid (COOH) in " ?name)
                (println " at atom " ?A1 ".")
;               (fail)
 )

((COOX ?name) if
                (has_COOX ?name ?A1 ?A2)
                (println "    Found Ester (COOX) in " ?name " at atom " ?A1 ".")
;               (fail)
 )

((N3 ?name) if
                (has_N3 ?name ?A1 ?)
                (println "    Found Amine (-NR2) in " ?name " at atom " ?A1 ".")
;               (fail)
 )

((NT ?name) if
                (has_NT ?name ?A1 ?)
                (print "    Found Tertiary Amine (-NR2) in " ?name)
                (println " at atom " ?A1 ".")
;               (fail)
 )

((NS ?name) if
                (has_NS ?name ?A1 ?)
                (print "    Found Secondary Amine (-NRH) in " ?name)
                (println " at atom " ?A1 ".")
;               (fail)
 )

((NP ?name) if
                (has_NP ?name ?A1 ?)
                (print "    Found Primary Amine (-NH2) in " ?name)
                (println " at atom " ?A1 ".")
;               (fail)
 )

; (has_CdC ?name ?A1 ?A2) - Searches for C=C's, and returns the name of
;    the molecule in which the bond was found ?name, and the two atom
;    numbers that encompass the bond.  The molecule name can be
;    supplied to reduce searching of the knowledge base.
                ((has_CdC ?name ?A1 ?A2) if
                        (molecule ?name ? ?atlist ?bdlist)
                        (println "Looking for a C=C in " ?name ":")
                        (finddb ?bdlist ?A1 ?A2)
                        (is_atom ?atlist "C" ?A1)
                        (is_atom ?atlist "C" ?A2)
                        (< ?A1 ?A2) ;Prevents reporting bonds twice due to flexibility
                 )              ;  of "finddb" routine.  Refer there for more info.

; (has_CtC ?name ?A1 ?A2) - Searches for -CC- 's, and returns the name of
;    the molecule in which the bond was found ?name, and the two atom
;    numbers that encompass the bond.  The molecule name can be
;    supplied to reduce searching of the knowledge base.
                ((has_CtC ?name ?A1 ?A2) if
                        (molecule ?name ? ?atlist ?bdlist)
                        (println "Looking for a -CC- in " ?name ":")
                        (finddb ?bdlist ?A1 ?A2)
                        (is_atom ?atlist "C" ?A1)
                        (is_atom ?atlist "C" ?A2)
                        (< ?A1 ?A2) ;Prevents reporting bonds twice due to flexibility
                 )              ;  of "finddb" routine.  Refer there for more info.

; (has_CtN ?name ?A1 ?A2) - Searches for -CN's, and returns the name of
;    the molecule in which the bond was found ?name, the atom number
;    of the carbon atom where it attaches to the molecule ?A1, and the atom
;    number of the nitrogen atom ?A2.  The molecule name can be supplied
;    to reduce searching of the knowledge base.
; NOTE - Don't need to use "(< ?A1 ?A2)" as noted in "finddb" predicate
;    because both atoms are not identical.  It does use the reversal
;    capability of "finddb" to find a tripple bond with a Nitrogen
;    atom.
                ((has_CtN ?name ?A1 ?A2) if
                        (molecule ?name ? ?atlist ?bdlist)
                        (println "Looking for a -CN in " ?name ":")
                        (finddb ?bdlist ?A1 ?A2)
                        (is_atom ?atlist "C" ?A1)
                        (is_atom ?atlist "N" ?A2)
                 )

; (has_Oath ?name ?A1 ?A2) - Searches for C-O-C's, and returns the name of
;    the molecule in which the ether bond was found ?name, one of the two
;    atom numbers that encompass the bond ?A2, and the number of the
;    oxygen atom ?A1.  The molecule name can be supplied to reduce
;    searching of the knowledge base.
                ((has_Oath ?name ?A1 ?A2) if
                        (molecule ?name ? ?atlist ?bdlist)
                        (println "Looking for a C-O-C in " ?name ":")
                        (is_atom ?atlist "O" ?A1)
                        (println "  Found Oxygen at atom " ?A1 ".  Is it O-X?")
                        (finddb ?bdlist ?A1 ?A2)
                        (is_atom ?atlist "C" ?A2)               ; Make sure it's a carbon
                        (println "  Single bond w/ 'O' found.  Look for bond w/another atom.")
                        (ofinddb ?bdlist ?A1 ?A2 ?A3)
                        (is_atom ?atlist "C" ?A3)               ; Make sure it's a carbon
                        (< ?A2 ?A3) ;here, two atoms on either side of 'O' are present
                 )              ;  this assures only one answer gets printed out.

; (has_OaH ?name ?A1 ?A2) - Searches for -OH's, and returns the name of
;    the molecule in which the bond was found ?name, the atom number
;    of the oxygen atom ?A1, and the atom number to which the -OH is
;    attached ?A2.
; NOTE - This finds ONLY C-O-H's (not P-O-H, etc)
;                - Excludes COOH from counting!
                ((has_OaH ?name ?A1 ?A2) if
                        (molecule ?name ? ?atlist ?bdlist)
                        (println "Looking for a -OH in " ?name ":")
                        (is_atom ?atlist "O" ?A1)
                        (println "  Found Oxygen at atom " ?A1 ".")
                        (finddb ?bdlist ?A1 ?A2)
                        (println "  Single bond w/ 'O' found.  Look for bond w/another atom.")
                        (is_atom ?atlist "C" ?A2)
                        (listcat (?A2) nil ?H_list)
                        (attached_H ?atlist ?bdlist ?A1 ?H_list)
                        (not (and                               ;Add below to exclude COOH from counting
                                (finddb ?bdlist ?A2 ?A3)
                                (is_atom ?atlist "O" ?A3)
                             )
                        )
                 )

; (has_CdO ?name ?A1 ?A2) - Searches for C=O's, and returns the name of
;    the molecule in which the bond was found ?name, the atom number
;    of the Carbon atom ?A1, and the atom number of the Oxygen atom ?A2.
; NOTE - Will report C=O's in COOH's and COOX's.
                ((has_CdO ?name ?A1 ?A2) if
                        (molecule ?name ? ?atlist ?bdlist)
                        (println "Looking for a C=O in " ?name ":")
                        (is_atom ?atlist "O" ?A2)
                        (println "  Found Oxygen at atom " ?A2 ".  Is it a C=O?")
                        (finddb ?bdlist ?A1 ?A2)
                        (is_atom ?atlist "C" ?A1)
;                       (not (and                       Add below to exclude COOH from counting
;                               (finddb ?bdlist ?A1 ?A3)
;                               (is_atom ?atlist "O" ?A3)
;                            )
;                       )
                 )

; (has_CHdO ?name ?A1 ?A2) - Searches for CH=O's (Aldehydes), and returns
;    the name of the molecule in which the bond was found ?name, the atom
;    number of the Carbon atom ?A1, and the atom number of the Oxygen
;            atom ?A2.  Remember, C-H's aren't present in the bond list, so for
;            Aldehydes there should be only one C-X.
; NOTE - Currently doesn't identify H2C=O!!!!!!!!!!
                ((has_CHdO ?name ?A1 ?A2) if
```

```
                    (molecule ?name ? ?atlist ?bdlist)
                    (println "Looking for an Aldehyde (CH=O) in " ?name ":")
                    (is_atom ?atlist "O" ?A2)
                    (println "  Found Oxygen at atom " ?A2 ".  Is it a C=O?")
                    (finddb ?bdlist ?A1 ?A2)
                    (is_atom ?atlist "C" ?A1)
                    (println "  Have C=O at atom " ?A1 ".  Is there a C-X bond?")
                    (findnb ?bdlist ?A1 ?A3)                        ;look for C-X at carbon ?A1.
                    (not (is_atom ?atlist "H" ?A3))                 ;don't want X = Hydrogen
                    (not (is_atom ?atlist "O" ?A3))                 ; or X = Oxygen (=Keytone)
                    (println "  Have C-X at atom " ?A1 "-" ?A3 ".  Is there a C-H?")
                    (listcat (?A3) nil ?H_list)
                    (attached_H ?atlist ?bdlist ?A1 ?H_list)        ;want to find C-H bond
            )

; (has_X2CdO ?name ?A1 ?A2) - Searches for X2C=O's (Keytones), and returns
;    the name of the molecule in which the bond was found ?name, the atom
;    number of the Carbon atom ?A1, and the atom number of the Oxygen
;            atom ?A2.
            ((has_X2CdO ?name ?A1 ?A2) if
                    (molecule ?name ? ?atlist ?bdlist)
                    (println "Looking for a Keytone (X2C=O) in " ?name ":")
                    (is_atom ?atlist "O" ?A2)
                    (println "  Found Oxygen at atom " ?A2 ".  Is it a C=O?")
                    (finddb ?bdlist ?A1 ?A2)
                    (is_atom ?atlist "C" ?A1)
                    (println "  Have C=O at atom " ?A1 ".  Is there a C-X bond?")
                    (findnb ?bdlist ?A1 ?A3)                        ;look for C-X at carbon ?A1.
                    (not (is_atom ?atlist "O" ?A3))                 ;don't want carboxylic acid
                    (not (is_atom ?atlist "H" ?A3))                 ;don't want Aldehyde
                    (println "  Have C-X at atom " ?A1 "-" ?A3 ".  Is there another C-X?")
                    (ofindnb ?bdlist ?A1 ?A3 ?A4)                   ;want to find another C-X
                    (not (is_atom ?atlist "O" ?A4))                 ;don't want carboxylic acid
                    (not (is_atom ?atlist "H" ?A4))                 ;don't want Aldehyde
                    (< ?A3 ?A4)                     ;don't want to report both X1-C-X2 and X2-C-X1
                    (println "  Yes, it's a different C-X, so it's an Keytone")
            )

; (has_COOH ?name ?A1 ?A2) - Searches for X2C=OOH's (Carboxylic acids),
;            and returns the name of the molecule in which the bond was found ?name,
;            the atom number of the Carbon atom ?A1, the atom number of the Carbonyl
;            Oxygen atom ?A2, but not the atom number of the other oxygen atom ?A3.
            ((has_COOH ?name ?A1 ?A2) if
                    (molecule ?name ? ?atlist ?bdlist)
                    (println "Looking for a Carboxylic acid (XC=OOH) in " ?name ":")
                    (is_atom ?atlist "O" ?A2)                       ;Look for C=O
                    (println "  Found Oxygen at atom " ?A2 ".  Is it a C=O?")
                    (finddb ?bdlist ?A1 ?A2)
                    (is_atom ?atlist "C" ?A1)
                    (println "  Have C=O at atom " ?A1 ".  Is there a C-O bond?")
                    (findnb ?bdlist ?A1 ?A3)                        ;Look for C-O at carbon ?A1.
                    (is_atom ?atlist "O" ?A3)
                    (println "  Have C-O at atom " ?A3 ".  Is there a -OH bond?")
                    (listcat (?A1) nil ?H_list)
                    (attached_H ?atlist ?bdlist ?A3 ?H_list) ;Look for OH at carbon ?A1.
            )

; (has_COOX ?name ?A1 ?A2) - Searches for X2C=OOX's (esters),
;            and returns the name of the molecule in which the bond was found ?name,
;            the atom number of the Carbon atom ?A1, the atom number of the Carbonyl
;            Oxygen atom ?A2, but not the atom number of the other oxygen atom ?A3.
; NOTE - Will report anhydrides as 2 COOX's....
            ((has_COOX ?name ?A1 ?A2) if
                    (molecule ?name ? ?atlist ?bdlist)
                    (println "Looking for a Ester (XC=OOX) in " ?name ":")
                    (is_atom ?atlist "O" ?A2)                       ; Look for O
                    (println "  Found Oxygen at atom " ?A2 ".  Is it a C=O?")
                    (finddb ?bdlist ?A1 ?A2)                        ; Look for =O
                    (is_atom ?atlist "C" ?A1)                       ; Is it a C=O?
                    (println "  Have C=O at atom " ?A1 ".  Is there a C-O bond?")
                    (findnb ?bdlist ?A1 ?A3)                    ;look for C-O at carbon ?A1.
                    (is_atom ?atlist "O" ?A3)
                    (ofindnb ?bdlist ?A1 ?A1 ?A4)               ;look for C-O-X at Oxygen ?A3.
                    (not (is_atom ?atlist "H" ?A4))             ;don't want X = Hydrogen
                    (not (is_atom ?atlist "O" ?A4))                 ; or X = Oxygen
            )

; (has_N3 ?name ?A1 ?) - Searches for Amines in molecule ?name, returning
;            the atom number of the Amine Nitrogen atom in ?A1.  The third parameter
;            is simply a place holder, and is not used.
            ((has_N3 ?name ?A1 0) if
                    (molecule ?name ? ?atlist ?bdlist)
                    (println "Looking for an Amine in " ?name ":")
                    (is_atom ?atlist "N" ?A1)                       ;Look for Nitrogen
                    (not (finddb ?bdlist ?A1 ?))                    ;Don't want any +N
                    (not (findtb ?bdlist ?A1 ?))                    ;Don't want any tb N
            )

; (has_NT ?name ?A1 ?) - Searches for Tertiary Amines (NR3) in molecule
;            ?name, returning the atom number of the Amine Nitrogen atom in ?A1.
;            The third parameter is simply a place holder, and is not used.
            ((has_NT ?name ?A1 0) if
                    (molecule ?name ? ?atlist ?bdlist)
                    (println "Looking for a Tertiary Amine (NR3) in " ?name ":")
                    (is_atom ?atlist "N" ?A1)                       ;Look for Nitrogen
                    (finddb ?bdlist ?A1 ?A2)                        ;Look for bond to molecule
                    (not (is_atom ?atlist "H" ?A2))                 ;Make sure it isn't a H
                    (println "  Found first bond N-R [" ?A1 "-" ?A2 "], other bond?")
                    (ofindnb ?bdlist ?A1 ?A2 ?A3)                   ;Look for a second bond
                    (< ?A2 ?A3)
                    (not (is_atom ?atlist "H" ?A3))                 ; that isn't a H
                    (println "  Found other bond N-R2 [" ?A1 "-" ?A3 "], attached H?")
                    (ofindnb ?bdlist ?A1 ?A2 ?A4)                   ;Look for a third bond
                    (not (== ?A3 ?A4))
                    (< ?A3 ?A4)
            )

; (has_N2 ?name ?A1 ?) - Searches for Secondary Amines (NR2H) in molecule
;            ?name, returning the atom number of the Amine Nitrogen atom in ?A1.
;            The third parameter is simply a place holder, and is not used.
            ((has_N2 ?name ?A1 0) if
                    (molecule ?name ? ?atlist ?bdlist)
                    (println "Looking for a Secondary Amine (NR2H) in " ?name ":")
                    (is_atom ?atlist "N" ?A1)                       ;Look for Nitrogen
                    (finddb ?bdlist ?A1 ?A2)                        ;Look for bond to molecule
                    (not (is_atom ?atlist "H" ?A2))                 ;Make sure it isn't a H
                    (println "  Found first bond N-R [" ?A1 "-" ?A2 "], other bond?")
                    (ofindnb ?bdlist ?A1 ?A2 ?A3)                   ;Look for a second bond
                    (< ?A2 ?A3)
                    (not (is_atom ?atlist "H" ?A3))                 ; that isn't a H
                    (println "  Found other bond N-R2 [" ?A1 "-" ?A3 "], attached H?")
                    (listcat (?A2) (?A3) ?H_list)
                    (attached_H ?atlist ?bdlist ?A1 ?H_list)        ;Verify other H attached
            )

; (has_NP ?name ?A1 ?) - Searches for Primary Amines (NRH2) in molecule
;            ?name, returning the atom number of the Amine Nitrogen atom in ?A1.
;            The third parameter is simply a place holder, and is not used.
; NOTE - Doesn't allow for Hydrogens to be specified!!
            ((has_NP ?name ?A1 0) if
                    (molecule ?name ? ?atlist ?bdlist)
                    (println "Looking for a Primary Amine (NRH2) in " ?name ":")
                    (is_atom ?atlist "N" ?A1)                       ;Look for Nitrogen
                    (findnb ?bdlist ?A1 ?A2)                        ;Look for bond to molecule
                    (not (is_atom ?atlist "H" ?A2))                 ;Make sure it isn't a H
                    (listcat (?A2) nil ?H_list)
                    (attached_H ?atlist ?bdlist ?A1 ?H_list)
                    (not (finddb ?bdlist ?A1 ?))                    ;Make sure db doesn't exist
            )


;   ***END OF assert
)

;                       **** File RING.HC ****                  February 26, 1993
(assert

; (aringq) - abolish all predicates in new.hc so you an try again
            ((aring) if
                    (abolish Chex) (abolish Benz) (abolish has-benz) (abolish Pyrdn)
                    (abolish has_Chex) (abolish has_Benz) (abolish has_Pyrdn)
                    (abolish all_rings) (abolish aromatic)
                    (abolish has_ring) (abolish find_ring)
                    (abolish arom) (abolish arom_db) (abolish arom_sb)
                    (abolish rotate_ring) (abolish ring_atom) (abolish low_mem)
                    (abolish aring) (consult ring.hc)
            )

            ((Chex ?name) if
                    (has_Chex ?name ?A1 ?A2)
                    (print **    Found Cyclohexane (Chex) in " ?name)
                    (println " at atom " ?A1 ": ring - " ?A2 ".")
                    (fail)
            )

            ((Benz ?name) if
                    (has_Benz ?name ?A1 ?A2)
        (print **    Found Benzene (Benz) in " ?name)
                    (println " at atom " ?A1 ": ring - " ?A2 ".")
                    (fail)
            )

            ((has-benz ?name) if
                    (has_Benz ?name ?A1 ?A2)
        (print **    Found Benzene (Benz) in " ?name)
                    (println " at atom " ?A1 ": ring - " ?A2 ".")
                    (fail)
            )

            ((Pyrdn ?name) if
                    (has_Pyrdn ?name ?A1 ?A2)
        (print **    Found Pyridine (Pyrdn) in " ?name)
                    (println " at atom " ?A1 ": ring - " ?A2 ".")
                    (fail)
```

```
            )

; (aromatic_ring ?atlist ?bdlist ?low_num ?r_list) - Finds an aromatic
;            ring in a molecule with atom and bond lists ?atlist and ?bdlist,
;            and returns the lowest atom number in ?low_num, and a list of the
;            atom numbers that make up the cyclohexane ring in ?r_list.
;            Case 1: Look for ring, see if its aromatic
;            Case 2: In smiles_str, if no rings are found, you need to have
;                    the predicate succeed anyway so that smiles_str will succeed.
; NOTE - Might not work for molecule with several different ring#!!!
    ((aromatic_ring ?atlist ?bdlist ?low_num ?rlist) if
                ;           (println "Looking for aromatic ring in " ?name ":")
          (has_ring ?atlist ?bdlist 6 "C" ?low_num ?rlist)
                            (lastonlist ?rlist ?lst)
                            (arom ?bdlist ?lst ?rlist)
                )
    ((aromatic_ring ? ? ? nil)          )           ;Added for generating SMILES strings

; (has_Chex ?name ?low_num ?r_list) - Finds a cyclohexane and returns the name
;            name of the molecule in ?name, the lowest atom number in ?low_num, and
;            a list of the atom numbers that make up the cyclohexane ring in ?r_list.
; NOTE - Might not work for a ring with only one or two double bonds, or a
;            molecule with several different rings!!!!
    ((has_Chex ?name ?low_num ?rlist) if
                            (molecule ?name ? ?atlist ?bdlist)
                ;           (println "Looking for Cyclohexane (Chex) in " ?name ":")
                            (has_ring ?atlist ?bdlist 6 "C" ?low_num ?rlist)
                            (lastonlist ?rlist ?lst)
                            (not (arom ?bdlist ?lst ?rlist))
                )

; (has_Benz ?name ?low_num ?r_list) - Finds a Benzene ring and returns the
;            name of the molecule in ?name, the lowest atom number in ?low_num, and
;            a list of the atom numbers that make up the cyclohexane ring in ?r_list.
; NOTE - Might not work for molecule with several different rings!!!!
    ((has_Benz ?name ?low_num ?rlist) if
                            (molecule ?name ? ?atlist ?bdlist)
                ;           (println "Looking for Benzene (Benz) in " ?name ":")
          (has_ring ?atlist ?bdlist 6 "C" ?low_num ?rlist)
                            (lastonlist ?rlist ?lst)
                            (arom ?bdlist ?lst ?rlist)
                )

; (has_Pyrdn ?name ?low_num ?r_list) - Finds a pyridine structure and returns
;            the name of the molecule in ?name, the atom number of the Nitrogen
;            atom in ?low_num, and a list of the atom numbers that make up the
;            pyridine ring in ?r_list.
; NOTE - Might not work for molecule with several different rings!!!!
    ((has_Pyrdn ?name ?low_num ?r_list) if
                            (molecule ?name ? ?atlist ?bdlist)
          (println "Looking for Pyridine (Pyrdn) in " ?name ":")
          (has_ring ?atlist ?bdlist 6 ? ?A1 ?rlist)
                            (println "      Found Cyclic structure: " ?rlist)
                            (is_atom ?atlist "N" ?atnum)
                            (println "      Nitrogen atom #" ?atnum ".")
                            (member ?atnum ?rlist)
                            (lastonlist ?rlist ?lst)
                            (arom ?bdlist ?lst ?rlist)
                            (rotate_ring ?atnum ?rlist ?r_list)
                )

; (all_rings ?name ?rsize_list ?arl ?allringlist) - Looks for rings in
;            molecule ?name for all sizes in the list ?rsize_list.  Builds a list
;            of information about those rings in ?arl, and when all sizes have been
;            investigated, returns this info in ?allringlist.  The format of the
;            ring information list is :
;                                    (Atom-type lowest-num-atom Aromatic #atoms-in-ring (ring-list))
;            where Atom-type - "C" if entirely a carbon ring, otherwise the symbol
;                                    of the single non-carbon element found in the ring.
;                  lowest-num-atom - lowest numbered atom in ring list, assumed to be
;                                    the "head" of the atom list.
;                  Aromatic - "A" if ring is aromatic, "N" otherwise.
;                  #atoms-in-ring - The number of atoms in the ring.
;                  (ring-list) - The list of atom numbers that make up the ring.
;            Case 1: Final case - No more searching for rings, return
;            Case 2: Looking for ring size of size ?fst.  Call itself recursively to
;                    see if there are any additional rings of that size.  Will fail
;                    for last ring found of size ?fst even though ring exists.
;            Case 3: Finds the last ring of size ?fst in molecule ?name.  Then goes
;                    on to look for rings of sizes remaining in ?rst.
;
    ((all_rings ? nil ?arlist ?arlist))
    ((all_rings ?name (?fst.?rst) ?arl ?arlist) if
                            (has_ring ?atlist ?bdlist ?fst ?atom ?lnum ?rl)
                            (not (member (? ? ? ? ?rl) ?arl))                           ;***
                            (aromatic     ?name ?rl ?arom)
      (all_rings ?name (?fst.?rst) ((?atom ?lnum ?arom ?fst ?rl).?arl) ?arlist)
    )                                                                                   ;***
    ((all_rings ?name (?fst.?rst) ?arl ?arlist) if
                            (has_ring ?atlist ?bdlist ?fst ?atom ?lnum ?rl)
                            (not (member (? ? ? ? ?rl) ?arl))                           ;***
                            (aromatic     ?name ?rl ?arom)
                            (all_rings ?name ?rst ((?atom ?lnum ?arom ?fst ?rl).?arl) ?arlist)
                )                                                                       ;***

; (aromatic ?bdlist ?rl ?Atom_Symbol) - Checks to see if the atoms in the ring
;            ?rl in the molecule ?name are aromatic.
;            Case 1: Ring is aromatic, so return "A".
;            Case 2: Ring is NOT aromatic, so return "N".
    ((aromatic ?bdlist ?rl "A") if
                            (lastonlist ?rl ?lst)
                            (arom ?bdlist ?lst ?rl)
                            (cut)
    )
    ((aromatic ? ? "N"))

; (has_ring ?atlist ?bdlist ?r_size ?atom ?low_num ?r_list) - Finds a cyclic
;            structure of size ?r_size, returns the name of the molecule in ?name,
;            the lowest atom number in ?low_num, and a list of the atom numbers
;            that make up the ring in ?r_list.  ?atom is passed to "ring_atom",
;            where it is used to check if a ring is made up of all "?atom" atoms.
; NOTE - To eliminate recursively searching around a ring structure, assign
;            the lowest number atom as the "head" atom.  The order of the ring is
;            determined by looking at the two atoms that are attached to the head
;            atom, and choosing the lower numerical value of the atom number as the
;            one to be second on the list.  This eliminates reporting two rings
;            because of counting around the ring structure in opposite directions.
    ((has_ring ?atlist ?bdlist ?r_size ?atom ?low_num ?r_list) if
                            (find_ring ?bdlist ?r_size nil ?list)
                            (low_num nil ?list ?low_num)                    ; Only want cases where
                ;           (println "      Low number on list " ?list " is (" ?low_num ").")
                            (firstonlist ?list ?low_num)                    ;   lowest atom number
                            (rotate_ring ?low_num ?list ?r_list)    ;   is first on atom list.
                            (lastonlist ?r_list ?lst)                       ; Don't want to print out
                            (seconlist ?r_list ?sec)                        ;   reverse order of ring.
                            (ring_atom ?atlist ?r_list ?atom)       ; Check if all atoms same type
                ;           (printf "      Last on list:" ?lst ", Second on list :" ?sec)
                ;           (println ", ?atom:" ?atom ".")
                            (< ?sec ?lst)
                ;           (println "      Finished Searching for ring: " ?r_list)
                )

; (find_ring ?bdlist ?ring_size ?rl ?rlist) - Check to
;            see if there is a ring of size ?ring_size in ?bdlist.  ?rl is used
;            in building the ring list, which is returned in ?rlist by Case 2.
;            Case 1 : Initial search for a ring.  The first bond in ?bdlist
;                     is chosen, its atoms added to the ring list, the ring count
;                     reduced by one, and recursively calls itself.
;            Case 2 : Final check for ring closure.  Determines the first and
;                     last members of the ring, and checks for a bond between them.
;            Case 3 : Add additional ring members.  Take the last member in the
;                     ring list, look for additional atoms that extend the ring.
    ((find_ring ?bdlist ?rs () ?rlist) if
                ;           (println "      Initial search for ring size " ?rs ".")
                            (is_bond ?bdlist ?A1 ?A2)
                            (listcat (?A1) (?A2) ?rl)
                            (:= ?rsiz [- ?rs 1])
                ;           (println "      Searching for ring : " ?rl)
                            (find_ring ?bdlist ?rsiz ?rl ?rlist)
                ;           (println "      Found ring : " ?rlist ", finished searching.")
                )
    ((find_ring ?bdlist 1 ?rlist ?rlist) if
                ;           (println "      Check for ring closure on ring " ?rlist ".")
                            (firstonlist ?rlist ?fst)
                            (lastonlist ?rlist ?lst)
                            (bond_type ?bdlist ?fst ?lst ?)
                ;           (println "      Have ring closure for ring : " ?rlist)
                )
    ((find_ring ?bdlist ?rs ?rl ?rlist) if
                ;           (println "      Have ring " ?rl ", search for ring size " ?rs ".")
                            (lastonlist ?rl ?lst)
                            (is_bond ?bdlist ?lst ?A1)
                            (not (member ?A1 ?rl))                  ; Make sure it's a different atom
                            (listcat ?rl (?A1) ?rlist)
                            (:= ?rsiz [- ?rs 1])
                ;           (println "      Searching for " ?rsiz " more members, have : " ?rlist)
                            (find_ring ?bdlist ?rsiz ?rlist ?rlist)
                )

; (arom ?bdlist ?lst ?r_list) - Determines whether the ring of atoms in
;            ?r_list is aromatic.  The last atom in ?r_list is passed in ?lst, and
;            the bond between atom ?lst and the first atom in ?r_list, ?fst, is
;            the first bond reviewed.  Then arom_sb and arom_db are called
;            alternately in a recursive manor until all the bonds between atoms in
;            the atom list have been checked.
;            Case 1: The first bond, between ?lst and ?fst, is a double bond.  The
;                    next bond between the head of the atom list and the second member
;                    of the atom list, must then be a single bond.
;            Case 2: The first bond, between ?lst and ?fst, is a single bond.  The
;                    next bond between the head of the atom list and the second member
;                    of the atom list, must then be a single bond.
;  arom_sb: The bond between the previous head of the bond list, ?lst, and
;                    the current head of the bond list, ?fst, must be a single bond.
;                    Hence, the next bond must be a double bond.
```

```
;  arom_db: The bond between the previous head of the bond list, ?lst, and
;                        the current head of the bond list, ?fst, must be a double bond.
;                        Hence, the next bond must be a single bond.
                ((arom ?bdlist ?lst (?fst.?rst)) if
                        (finddb ?bdlist ?lst ?fst)
                ;       (println "      Found initial SB, looking for DB in " ?rst ".")
                        (arom_db ?bdlist ?fst ?rst)
                )
                ((arom ?bdlist ?lst (?fst.?rst)) if
                        (finddb ?bdlist ?lst ?fst)
                ;       (println "      Found initial DB, looking for SB in " ?rst ".")
                        (arom_sb ?bdlist ?fst ?rst)
                )
                ((arom_sb ?bdlist ?lst (?fst.?rst)) if
                        (finddb ?bdlist ?lst ?fst)
                ;       (println "      Found aromatic SB, looking for DB in " ?rst ".")
                        (arom_db ?bdlist ?fst ?rst)
                )
                ((arom_sb ? ? nil))
                ((arom_db ?bdlist ?lst (?fst.?rst)) if
                        (finddb ?bdlist ?lst ?fst)
                ;       (println "      Found aromatic DB, looking for SB in " ?rst ".")
                        (arom_sb ?bdlist ?fst ?rst)
                )
                ((arom_db ? ? nil))

; (rotate_ring ?low_num ?list ?r_list) - Takes the current ring list, ?list,
;             and rotates the list so that the lowest atom number, ?low_num, is at
;             the head of the list, and returns it in ?r_list.
;             Case 1 : final case - The lowest atom number is at the head of the list,
;                      so return this list.
;             Case 2 : The current head is not the lowest atom number, so attach it
;                      to the end of the remainder of the list and recursively call
;                      itself to finish up.
                ((rotate_ring ?low_num (?low_num.?rst) (?low_num.?rst)) if
                ;       (println "      Finished list is [" (?low_num.?rst) "].")
                        (cut)           )
                ((rotate_ring ?low_num (?num.?rst) ?r_list) if
                        (listrat ?rst (?num) ?list)
                ;       (println "              List was [" (?num.?rst) "], is now [" ?list "].")
                        (rotate_ring ?low_num ?list ?r_list)      )

; (ring_atom ?atlist ?r_list ?atom) - Check if the ring list ?r_list is made
;             up of all "?atom" atoms.  Typically, "C" will bew specified if looking
;             for a carbon ring and "?" will be used otherwise.  In the case where
;             the ring is made up of all carbon atoms except one, ?atom can be used
;             to pass that one atom back (see Case 4).
; NOTE - "ring atom" will fail if there are two or more non-carbon atoms that
;             are NOT alike.  In Case 2 and 4, if ?atom is not yet unified, the
;             predicate "is_atom" will unify it.  If ?atom has already been unified,
;             the atom of the current atom number must be the same element as found
;             previously in case 4.
;             Case 1 : Final case - Last item on ring list is a carbon atom, return
;                      "C" in ?atom.  Will fail if it isn't a carbon atom!  Also, if ?
;             Case 2 : Final case - Since the last item on ring list is not a carbon,
;                      return the atom symbol in ?atom.
;             Case 3 : Check the first member of the ring list to see if it is a carbon
;                      atom, then recursively call itself to process the remaining list.
;             Case 4 : Error case - A non-carbon atom was found in the ring list,
;                      return the atom in ?atom.
                ((ring_atom ?atlist (?fst.nil) "C") if
                ;       (println "      Case 1: Is the last atom #" ?fst " a Carbon?")
                        (is_atom ?atlist "C" ?fst)
                ;       (println "                                                              ***Yes!")
                        (cut)
                )
                ((ring_atom ?atlist (?fst.nil) ?atom) if
                        (is_atom ?atlist ?atom ?fst)
                ;       (println "      Case 2: The last atom #" ?fst " is a " ?atom ".")
                        (cut)
                )
                ((ring_atom ?atlist (?fst.?rst) ?atom) if
                ;       (println "      Case 3: Is atom #" ?fst " a Carbon?")
                        (is_atom ?atlist "C" ?fst)
                ;       (println "                                                              ***Yes!, pass on
list " ?rst)
                        (cut)
                        (ring_atom ?atlist ?rst ?atom)
                )
                ((ring_atom ?atlist (?fst.?rst) ?atom) if
                ;       (println "      Case 4: Atom #" ?fst "-")
                        (is_atom ?atlist ?atom ?fst)
                ;       (println "                  is a -" ?atom ".  Found non-carbon, list :" ?rst)
                        (cut)
                )

; (low_mem ?cur_low ?list ?lowest) - returns the lowest member found in the
;             list ?list in ?lowest.  ?cur_low is used to pass the current lowest
;             member as the function recursively calls itself.
;             Case 1 : initial case, where there is no current lowest number ?cur_low
;             Case 2 : finial case - when the end of the list is encountered, the
;                      answer is returned in ?lowest.
;             Case 3 : The first member in the list is less than the current lowest
;                      number, so ?cur_low stays the current lowest number.
;             Case 4 : The first member in the list is greater than the current
;                      lowest number, so it is passed as the current lowest number.
                ((low_mem nil (?low.?list) ?lowest) if
                ;       (println "      Case 1: Low=" ?low ", list=" ?list ".")
                        (cut)
                        (low_mem ?low ?list ?lowest)
                )
                ((low_mem ?cur_low nil ?cur_low) if
                ;       (println "      DONE with list, lowest number is " ?cur_low ".")
                ;       (println "        Case 2: cur-low=" ?cur_low ".")
                        (cut)
                )
                ((low_mem ?cur_low (?fst.?list) ?lowest) if
                ;       (print "        Case 3: cur-low=" ?cur_low ", fst=" ?fst)
                ;       (println ", list=" ?list ".")
                        (< ?cur_low ?fst)
                        (low_mem ?cur_low ?list ?lowest)
                )
                ((low_mem ?cur_low (?fst.?list) ?lowest) if
                ;       (print "        Case 4: cur-low=" ?cur_low ", fst=" ?fst)
                ;       (println ", list=" ?list ".")
                        (> ?cur_low ?fst)
                        (low_mem ?fst ?list ?lowest)
                )

;   ***END OF assert
)

;                               **** File SMILES.HC ****                    February 26, 1993
(assert

; (asmiles) - abolish all predicates in smiles.hc so you an try again
                ((asmiles) if
                        (abolish get_smiles)        (abolish check_ss)
                        (abolish getall_smiles)     (abolish get_frag_ss)
                        (abolish asavall_smiles)    (abolish smiles_efr)
                        (abolish atsmiles)                          (abolish bdsmiles)
                        (abolish cnt_Hatoms)
                        (abolish cnt_atom)                          (abolish cnt_bond)
                        (abolish conv_atom)         (abolish conv_bond)
                        (abolish atsubscript)       (abolish write_field)
                        (abolish asmiles)                           [consult smiles.hc)
                )

; (get_smiles ?name) - Convert the internal molecule representation into
;             a format that can be read by the CONVB program that generates Smiles
;             strings, and save it to the file TRST.CVT.  Run DO-CONVB, read in the
;             smiles string, and add it to the Molecule Knowledge base.  Returns the name
;             of the molecule in ?name if it isn't specified.  DO-CONVB.BAT transfers
;             the mol file TRST.CVT to the PC, runs CONVB, then transfers the
;             resulting file TRST.SS which contains the smiles string back to read in.
                ((get_smiles ?name) if
                        (molecule ?name ? ?atlist ?bdlist)
                        (smiles_str ?name ? ?atlist ?bdlist "test.cvt")
                        (print "\nRun DO-CONVB.BAT now to get the SMILES string for ")
                          (println "the molecule\n              " ?name ".  Then")
                          (print "                    hit RETURN when ready to proceed.....")
                        (getline stdin ?)
                        (open test.ss read smilestr)
                        (getline smilestr ?ss)
                        (close smilestr)
                        (cut)
                        (check_ss ?name ?ss)                    ;see if a ss already exists
                        (retract ((molecule ?name ? ?atlist ?bdlist)) )
                        (assert ((molecule ?name ?ss ?atlist ?bdlist)) )
                        (cut)
                )

; (check_ss ?name ?ssnew) - Check to see if the molecule ?name already
;             has an existing Smiles string within the runtime molecule knowledge base.
;             ?ssnew contains the new Smiles string just determined.
;             Case 1: Molecule ?name doesn't have a Smiles string in the runtime KB.
;             Case 2: Molecule ?name has a Smiles string, so ask the user if they
;                     would like to replace the currently existing Smiles string with
;                     the new string ?ssnew.  If so, 'Y' or 'y' must be typed to
;                     succeed so that the existing molecule to be retracted.
; NOTE - If the user doesn't want it replaced, case 2 will fail.  IF the
;                     system comes back again and asks the question, than duplicate
;                     entries for that molecule exist WHICH MUST BE CORRECTED.
                ((check_ss ?name ?ss) if
                        (molecule ?name ss ? ?)    )
                ((check_ss ?name ?ssnew) if
                        (molecule ?name ?ss ? ?)
                        (print "\nNote that the Molecule " ?name " already has a Smiles ")
                          (print "string\n      [" ?ss "].  Would you like to replace it ")
                          (print "with\n      [" ?ssnew "]? (Y or N, N by default)  > ")
                        (read ?Char)
                        (= ?ans (strcase ?Char upper))
                        (cut)                       ; cut to prevent backtracking
```

```
                    (= *Y* ?ans)
                    (println *          Replacing Molecule * ?name *.....*)
               )

; (getall_smiles ?name) - Determine the 4 fragments of the molecule around
;            the chiral centers, find the smiles string for the fragments ONLY, and
;            add them to the runtime molecule fragment Knowledge base.  Returns the name
;            of the molecule in ?name if it is not specified.  If you want to save
;            the fragment information to the molfrag knowledge base, use "saveall_smiles".
; NOTE - The smiles string must already be specified for it to succeed
               ((getall_smiles ?name) if
                                                ;***** Get and check Molecule for Chirality
                    (molecule ?name ?ss ?atlist ?bdlist)
                    (not (= ?ss "ss"))          ;Make sure SS exists for ?name
                    (find_chiral ?atlist ?cnum)
                    (> ?cnum 0)                 ;Chiral atom number must be > 0
                    (check_molfrag ?name ?ss)   ;Check if frag info already exists
                                                ;***** Saving fragment files
                    (print *\nChiral atom number * ?cnum * found in * ?name *.  Please*)
                      (print *\n     enter a file name for saving Molecule *)
                      (print *fragment files (NO EXT): -> *)
                    (getline stdin ?fn)
                    (:= ?fn1 [strcase ?fn upper])
                    (:= ?fname [strcat ?fn1 *.CV*])
;                   (println *          Fragment file name: [* ?fname *].*)
                    (fragment_mol ?cnum ?atlist ?bdlist ?flist)
                    (println * The fragment list for * ?name * is\n     [* ?flist *]*)
                    (save_frag_mol ?name ?atlist ?bdlist ?flist ?fname 1)
                                                ;***** Getting SMILRS strings for fragments
                    (print *\nRun CONVRALL.BAT now with the fragment file [* ?fn1 *] to *)
                      (print *get the SMILRS string\n      for the fragments of *)
                      (print ?name *.  Hit\n     return when ready to proceed *)
                      (print *[i.e. CONVRALL * ?fn1 *].....*)
                    (getline stdin ?)
                    (sys *rename [stauffers]*.ss* [stauffers.fgroup]*.**)
                    (:= ?ssfname [strcat ?fn1 *.SS*])
;                   (println *\n          SMILRS string file name: [* ?ssfname *].*)
                    (get_frag_ss ?ssfname 1 ?ss1)       (get_frag_ss ?ssfname 2 ?ss2)
                    (get_frag_ss ?ssfname 3 ?ss3)
                    (or (
                              (lastonlist ?flist ?lst)
                              (not (= ?lst nil))
                              (get_frag_ss ?ssfname 4 ?ss4)          )
                        (
;                             (println *     Last fragment was Hydrogen, NO Smiles string...*)
                              (:= ?ss4 **)              )
                    )           ;*** End OR
                    (print *  The smiles string list is\n          [*)
                    (println ?ss1 * * ?ss2 * * ?ss3 * * ?ss4 *]*)
                    (cut)
                                                ;***** asserting fragment SMILRS strings to mol-frag KB
                    (print *  Making sure the information doesn't exist in the runtime *)
                      (println *(fragment knowledge base...*)
                    (not (molfrag ?ss ?ss1 ?ss2 ?ss3 ?ss4) )
                    (assert ((molfrag ?ss ?ss1 ?ss2 ?ss3 ?ss4))         )
                    (cut)
                                                ;***** Deleting fragment files from SMILRS generation
;??? ADD THIS BACK IN FOR FINAL VERSION, removed to build KB
;                   (print *\nDo you want to delete the files * ?fname *# and *)
;                     (print ?ssfname *# that were used\n     to generate the Smiles *)
;                     (print *strings?  [Y-yes (default), N-no]  Y/N -> *)
;                   (read ?Char2)
;                   (:= ?ans2 [strcase ?Char2 upper])
;                   (not (= *N* ?ans2))
                    (:= ?sys1 [strcat *Delete * ?fname*])
                    (:= ?sys2 [strcat ?sys1 **;**])
                    (print *          * ?sys2 *, used for generating Smiles strings.*)
                    (sys ?sys2)   ;NOTE - Prints output to the next line automatically
                    (:= ?sys3 [strcat *Delete * ?ssfname*])
                    (:= ?sys4 [strcat ?sys3 **;**])
                    (print *          * ?sys4 *, used for generating Smiles strings.*)
                    (sys ?sys4)   ;NOTR - Moves output to the next line automatically
               )                                ;**** END ((getall_smiles ?name)

; (get_frag_ss ?fname ?fnum ?ss) - Opens and reads the Smiles string
;            from the base filename ?fname and returns it in ?ss.  ?fnum will
;            be a number from 1 to 4 and be combine with ?fname to read in the
;            smiles strings.
; NOTE - If the open statement fails, Prolog will echo an error *no such
;                   file or directory*.
               ((get_frag_ss ?fn ?fnum ?ss) if
                    (:= ?fname [strcat ?fn [numtostr ?fnum 0]])         ;Build SS filename
                    (open ?fname read ssfile)
;                   (println *     Getting smiles string from [* ?fname *]....*)
                    (getline ssfile ?ss)
                    (close ssfile)
;                   (println *          Smiles String is [* ?ss *].*)
                    )

; (saveall_smiles ?name) - Save the fragment information for ?name.  It
;            assumes that you want to save the fragment information to the
;            *mol-frag* knowledge base, and does NOT DO ANY CHECKING.  It should
;            only ever be done once after *getall_smiles* is called to determine
;            the fragments and save them to the runtime *mol-frag* knowledge base.
               ((saveall_smiles ?name) if
                    (molecule ?name ?ss ? ?)    (molfrag ?ss ?ss1 ?ss2 ?ss3 ?ss4)
                    (println *  Saving * ?name * to Fragment knowledge base... *)
                    (open *mol-frag.kb* append mfrag_kb)
                    (writeterm mfrag_kb ?ss)                                    ;Write molecule s-str to kb
                         (write mfrag_kb * *)   (writeterm mfrag_kb ?ss1)
                         (write mfrag_kb * *)   (writeterm mfrag_kb ?ss2)
                         (write mfrag_kb * *)   (writeterm mfrag_kb ?ss3)
                         (write mfrag_kb * *)   (writeterm mfrag_kb ?ss4)
                              (writeln mfrag_kb **)
                    (println *    Saved * ?name * to Fragment knowledge base... *)
                    (close mfrag_kb)
                    (cut)
               )

; (smiles_str ?name ? ?atlist ?bdlist ?filename) - Convert the internal
;            representation of the molecule (?name ?ss ?atlist ?bdlist) to a
;            form that can be read by the Smiles string conversion program CONVB
;            which came with the program SPARC, and save it to the file ?filename
;  NOTE - ?name is specified for identification reasons only in the
;            file generated for CONVB.
               ((smiles_str ?name ? ?atlist ?bdlist ?filename) if
                    (cnt_Hatoms ?atlist ?Hnum)
                    (cnt_atom ?atlist ?atnum)
                    (cnt_bond ?bdlist ?bdnum)
                    (open ?filename write smiles)
                    (write_field smiles          [- ?atnum ?Hnum] 5)
                    (write smiles * ATOMS *)
                    (write_field smiles [- ?bdnum ?Rnum] 5)
                                   (writeln smiles * BONDS    0 CHARGES * ?name)
;                   (print *  * [- ?atnum ?Rnum] * ATOMS     *   [- ?bdnum ?Hnum])
;                                  (println * BONDS    0 CHARGES * ?name)
                    (aromatic_ring ?atlist ?bdlist ? ?rlist)
;                   (println *     Ring list is {* ?rlist *}.*)
                    (atsmiles ?atlist ?bdlist 0 ?rlist ?Hlist)
;                   (println *     Hydrogen list is {* ?Hlist *}.*)
                    (bdsmiles ?bdlist 0 ?rlist ?Hlist)
                    (writeln smiles **)
                    (close smiles)
                    (cut)
               )

; (atsmiles ?atlist ?bdlist ?atnum ?rlist ?Hlist) - Write the atom list
;            ?atlist to the stream *smiles* in the format required by SPARC's MOL
;            files.  ?atnum is the current atom number being written out.  It will
;            agree with the atom number found in the atom list if there are NO
;            Hydrogens found.  ?Hlist, a list of pairs consisting of the actual
;            atom number and the atom number assigned after removing hydrogens,
;            is the Hydrogen list that is built up after all Hydrogens are found
;            as the function comes out of its recursive calls.
;            Case 1: There are no more atoms to process, so return the list of
;                    Hydrogen atoms found in the last parameter
;            Case 2: The current atom is a Hydrogen atom, so eliminate it by not
;                    incrementing the atom counter, and go on
;            Case 3: Base case.  Write out atom information for SPARC MOL file.
               ((atsmiles nil ? ? ? nil))
               ((atsmiles ((atom ?atnum *H* ?) . ?rst) ?bdlist ?num ?rlist ?Hlist) if
;                   (println *   Hydrogen found at atom * ?atnum *.  Eliminating it...*)
                    (atsmiles ?rst ?bdlist ?num ?rlist ?Hlist)
               )
 ((atsmiles ((atom ?atnum ?elem ?).?rst) ?bdlist ?num ?rlist (?lst.?Hlist)) if
                    (:= ?num [+ ?atnum 1])
               ;???  (print *    * ?num * * ?elem)
                    (write_field smiles ?num 5)
                    (write smiles * * [strcase ?elem upper])            ;CONVB wants upper case
                    (atsubscript smiles ?atnum ?elem ?bdlist ?rlist)
                    (listcat (?atnum) (?num) ?lst)                      ;want actual atom number
;                   (print *  **** At# {* ?atnum *}, SS# {* ?num)
;                   (println *}, lcat {* ?lst *}, Hl {* ?Hlist *}*)
                    (atsmiles ?rst ?bdlist ?num ?rlist ?Hlist)
               )

; (bdsmiles ?bdlist ?bdnum ?rlist ?Hlist) - Write the bond list ?bdlist
;            to the stream *smiles* in the format required by SPARC's MOL files.
;            ?bdnum counts the number of the bond being written to the file.
;            Case 1: Final case - No more bonds on the bond list ?bdlist.
;            Case 2: If either ?at1 or ?at2 is a Hydrogen atom, conv atom will
;                    fail, causing Case 2 to fail, and control will pass to Case 3.
;            Case 3: Current bond is with a Hydrogen, so remove it from the bond
;                    list and go on.
               ((bdsmiles nil ? ? ?))
               ((bdsmiles ((? ?at1 ?at2 ?bd) . ?rst) ?bdnum ?rlist ?Hlist) if
                    (conv_atom ?at2 ?atm2 ?Hlist)
                    (conv_atom ?at1 ?atm1 ?Hlist)
                    (:= ?num [+ ?bdnum 1])
                    (conv_bond ?bd ?bnd ?at1 ?at2 ?rlist)
;                   (print *    * ?num *    * ?atm1 *     * ?atm2 *   * ?bnd)
;                   (println *   ***old Info*** * ?at1 *     * ?at2)
                    (write_field smiles ?num 5)
                    (write smiles * *)
```

```
                        (write-field-smiles ?atm1 %)
                        (write-smiles " ")
                        (write-field-smiles ?atm2 %)
                        (writeln-smiles "  " ?bnd)
                        (bdsmiles ?rst ?num ?rlist ?Hlist)
                )
                ((bdsmiles ((? ?at1 ?at2 ?bd) . ?rst) ?bdnum ?rlist ?Hlist) if
                ;               (println "    Passing Hydrogen list {" ?Hlist "}.")
                ;               (println "    Get rid of bond {" ?at1 " " ?at2 " " ?bd "}.")
                                (bdsmiles ?rst ?bdnum ?rlist ?Hlist)
                )

; (cnt_Hatoms ?atlist ?Hnum) - Counts the number of Hydrogen atoms found
;               in ?atlist and returns it in ?Hnum.  The actual counting is done as
;               the predicates come back from recursing after the end of the atom
;               list is found.
;               Case 1: End of ?atlist found.
;               Case 2: A Hydrogen atom found, so increase count.
;               Case 3: A NON-Hydrogen atom found.  Leave count as is.
                ((cnt_Hatoms nil 0))
                ((cnt_Hatoms ((atom ? "H" ?) . ?rst) ?Hnum) if
                                (cnt_Hatoms ?rst ?num)
                                (:= ?Hnum [+ ?num 1])       )
                ((cnt_Hatoms (? . ?rst) ?Hnum) if
                                (cnt_Hatoms ?rst ?Hnum)      )

; (cnt_atom ?atlist ?atnum) - Determine the number of atoms contained
;               in the atom list ?atlist by returning the number of the last atom
;               found in the atom list in variable ?atnum.
                ((cnt_atom nil 0))
                ((cnt_atom ((atom ? ? ?) . ?rst) ?atnum) if
                                (cnt_atom ?rst ?num)
                                (:= ?atnum [+ ?num 1])      )

; (cnt_bond ?bdlist ?bdnum) - Count the number of bonds contained in
;               the bond list ?bdlist and return the number in ?bdnum.  The actual
;               counting is done as the predicates come back from recursing after
;               the end of the atom list is found.
                ((cnt_bond nil 0))
                ((cnt_bond ((? ? ? ?) . ?rst) ?bdnum) if
                                (cnt_bond ?rst ?num)
                                (:= ?bdnum [+ ?num 1])      )

; (conv_atom ?at ?atm ?newlist) - ?newlist contains a list where each member
;               contains two numbers, the first is the first number is the original atom
;               number from ?atlist, and the second is the new number.  ?at, the atom
;               number from the atom list, is looked up in ?newlist to see what the
;               corrected atom number, ?atm, is.
;               Case 1: ?at found in ?newlist, so return the new atom number in ?atm.
;               Case 2: ?at NOT found in ?newlist, so recursively look at remaining list.
;  NOTE - If this predicate fails, it means that the atom number, ?at, did
;                       not have an entry in the atom new number list ?newlist.
;                  - Can be used to remove hydrogen atoms from the atom list.  The
;                       predicate atmsmiles constructs the list as it processes the atom
;                       list, and the passes it to bdsmiles.  When no Hydrogen atoms are
;                       present, ?at and ?atm should be identical.
                ((conv_atom ?at ?atm ((?at ?atm) . ?)) )
                ;               (println "Found Atom {" ?at "} renumbered {" ?atm "}.")
                ((conv_atom ?at ?atm (?b.?newlist)) if
                ;               (print "Passed Atom {" ?at "} since it didn't match {" ?b "}")
                ;               (println "      and list {" ?newlist "}.")
                                (conv_atom ?at ?atm ?newlist)           )

; (conv_bond ?bd ?bnd ?atm1 ?atm2 ?rlist) - To convert bond-type numbers to
;               words, as required by SPARC's MOL file format.  Typically, the ?bd is
;               specified, and ?bnd is used to return the answer.  ?atm1, ?atm2, and
;               ?rlist are used when an aromatic carbon is found.
;               Case 1: The atoms in the current bond are checked if they exist in the
;                       bond list.  If so, it must be an aromatic bond.
;               Case 2-4: Converts the bond-type number to their written equivalent.
                ((conv_bond ? AROMATIC ?atm1 ?atm2 ?rlist) if
                ;               (println "      Is atom " ?atm1 " and "?atm2" in list " ?rlist ".")
                                (member ?atm1 ?rlist)
                                (member ?atm2 ?rlist)
                                (cut)           )
                ((conv_bond 1 SINGLE ? ? ?))
                ((conv_bond 2 DOUBLE ? ? ?))
                ((conv_bond 3 TRIPLE ? ? ?))

; (atsubscript ?stream ?atnum ?elem ?bdlist ?rlist) - To output the atom
;               subscripts required by SPARC's MOL file format.  Here, don't follow the
;               note found in findb to include a (< ? ?).  This is because the order of
;               that the atoms are listed in the bond list doesn't matter, since you
;               aren't trying to find multiple occurances.
;               Case 0: If current atom is a halogen, use no subscript.
;               Case 1: If the current atom number ?atnum is a member of the ring
;                       list ?rlist, use a subscript "AR" to indicate an Aromatic
;                       carbon.  The ring list is determined by calling "has_Renz".
; NOTE - Currently only handles ONE ring per molecule!!!!!!
;               Case 2: Use a subscript "1" if the atom has a triple bond
;               Case 3: Use a subscript "2" if the atom has a double bond
;               Case 4: Use a subscript "3" if the atom has a single bond
;               ((atsubscript ?stream ?atnum ?elem ?bdlist ?rlist) if
;                               (is_atom ?atlist "H" ?atnum)            NOTE this requires atlist!
;               ;               (println "      Ring list is {" ?rlist "}, atom number is:" ?atnum ".")
;                               (writeln ?stream "PL3")
;               ;               (println "PL3")  For NO2
;                               (cut)
;               )???
                ((atsubscript ?stream ? ?elem ? ?) if
                                (or             (= "Cl" ?elem)              (= "CL" ?elem)
                                                (= "F" ?elem)
                                                (= "Br" ?elem)              (= "BR" ?elem)
                                                (= "I" ?elem)               )
                                (writeln ?stream "")
                ;               (println "      Found Halogen:" ?elem " at atom number:" ?atnum ".")
                                (cut)
                )
                ((atsubscript ?stream ?atnum ? ?bdlist ?rlist) if
                                (member ?atnum ?rlist)
                ;               (println "      Ring list is {" ?rlist "}, atom number is:" ?atnum ".")
                                (writeln ?stream "AR")
                ;               (println "AR")
                                (cut)
                )
                ((atsubscript ?stream ?atnum ? ?bdlist ?) if
                                (findrb ?bdlist ?atnum ?at2)
                                (writeln ?stream "1")
                ;               (println "1")
                                (cut)
                )
                ((atsubscript ?stream ?atnum ? ?bdlist ?) if
                                (findb ?bdlist ?atnum ?at2)
                                (writeln ?stream "2")
                ;               (println "2")
                                (cut)
                )
                ((atsubscript ?stream ? ? ? ?) if
                                (writeln ?stream "3")
                ;               (println "3")
                                (cut)
                )

; (write_field ?stream ?num ?field_size) - Writes the number ?num right
;               justified within a field of size ?fieldsize, using spaces to pad out
;               the results.
;               Case 1: for numbers taking up 3 positions.
;               Case 2: for numbers taking up 2 positions.
;               Case 3: for numbers taking up 1 position.  (NO checking is done here)
                ((write_field ?stream ?num ?field_size) if
                                (> ?num 99)
                ;               (print "{" ?num ">99}.")
                                (writen ?stream [- ?field_size 3] " ")
                                (write ?stream ?num)
                                (cut)
                )
                ((write_field ?stream ?num ?field_size) if
                                (> ?num 9)
                ;               (print "{" ?num ">9}")
                                (writen ?stream [- ?field_size 2] " ")
                                (write ?stream ?num)
                                (cut)
                )
                ((write_field ?stream ?num ?field_size) if
                ;               (print "{" ?num "<9}")
                                (writen ?stream [- ?field_size 1] " ")
                                (write ?stream ?num)
                                (cut)
                )


;   ***END OF assert
)

;                               **** File SPARC.HC ****                 February 26, 1993
(assert

; (asparc) - abolish all predicates in sparc.hc so you an try again
                ((asparc) if
                                (abolish update_fq-valu)                (abolish new_valu_kb)
                                (abolish getall_sparc)                  (abolish sparcfreq_batch)
                                (abolish checkfreq)
                                (abolish getsparc_info)                 (abolish getsparc_record)
                                (abolish savef_value)                   (abolish sparc_batch)
                                (abolish load_sparc)                    (abolish ld_sparc)
                                (abolish get_term)                                      (abolish getinfo)
                                (abolish clear_line)
                                (abolish asparc)                                        (consult sparc.hc)
                )

; (update_fq-valu ?property) - Replace all of the values for one of the
;               property Knowledge bases.  ?property is the name of the property,
;               and can be any one of the following:
;                                       pol, Average molecular polarizability
```

```
;                                       alpha, H-bonding electron donating capabilities
;                                       beta, H-bonding electron accepting capabilities
;                                       vol, Molecular volume
;           This predicate handles first writing SPARC Batch Mode input file
;           BATCH.DAT with all needed info, then pauses to let the user run SPARC
;           to generate all values, then reads in the data and creates the new KB.
            ((update_fg-valu ?property) if
               ;*** Output all fragment information to Batch Mode for SPARC
                        (open "batch.dat" write sparc)
                        (or             (  (fg-valu ?ee?frag_pol ?) ;use existing prop 2get all frags
                                           (not (= ?ee?frag ""))                ;make sure it isn't just "H"
                                           (writeln sparc "['/25'].\n" ?ee?frag "'.\n" ?property ".")
                                           (fail)     )
                                        (  (println "  Done writing.")            )
                        )
                        (close sparc)                           (cut)
               ;*** Input results of new SPARC fragment values AND
               ;*** Write new SPARC fragment values to fragment value KB file
                        (print "\nRun TO-SPARC.BAT now, followed by SPARC's Batch ")
                          (print "mode.  Hit return\n      when ready to proceed ")
                          (print "(i.e. TO SPARC).....")
                        (getline stdin ?) (println " ")
                        (= ?fv_filename (strcat "fv_" (strcat ?property ".kb")))
                        (println " " Writing information to file [" ?fv_filename "].")
                        (open "batch.out" read sparckb)
                        (open ?fv_filename write fragval_kb)
                        (writeln fragval_kb "\22\22 " ?property " 0");Write out double quote
                        (getline sparckb ?a)                            ; for pol SS and a Zero
                        (get_term ?a)                                   ;Find "BATCH MODE OUTPUT"
                        (new_valu_kb ?property)                         ;Get and write all the data
                        (close sparckb)
                        (close fragval_kb)
            )

; (new_valu_kb ?property) - For reading and writing each record of data for
;           updating the fragment value knowledgebase.  Reads the SPARC Batch Mode
;           Output file that contains the data, and writes it to the fg-value KB.
;           Case 1: Read info from SPARC.OUT, write it to FV_PROP.KB
;           Case 2: Needed so that after EOF is reached, predicate will succeed.
; NOTE - All files must be open before calling this predicate.
            ((new_valu_kb ?property) if
                        (getsparc_record ?eef ?prop ?value)     ;Read fragment data
                        (writeterm fragval_kb ?eef)             ;Write fragment e-str to KB
                        (writeln fragval_kb " " ?property " " ?value)
                        (cut)                                   ;Write property & value to new fv KB file
                        (new_valu_kb ?property)
            )
            ((new_valu_kb ?) if
                        (println "  Done writing.")                      )

; (getall_sparc ?name) -  Loads results of property estimation calculated
;           using the program SPARC from the file ?output.  It is currently (5/92)
;           written for 3 properties (pol, alpha, beta) for each of the four
;           possible fragments around the chiral center.  If one of the fragments
;           is a Hydrogen, it's smiles string is not output, as it is handled as
;           a special case.  It first checks to make sure that there is not
;           already values stored for the molecule.
; NOTE - This procedure assumes a specific output format from SPARC, namely:
;                                       *** BATCH MODE OUTPUT ***
;
;                         *** Processing Molecule Number 1
;                         *** Smiles String: Cl-c(ccc1)cc1
;                         *** Property Type: Solubility
;                         *** Solvent Mixture: water
;                         *** Temperature: 25
;                         *** Observed Value's: [/25]
;                         *** Calculated Value: -4.276654759
;
;                         *** Processing Molecule Number 2
;                         *** Smiles String: Cl-c(ccc1)cc1
; NOTE   the escape sequences placed in the SPARC output file MUST BE
;                       REMOVED PRIOR TO READING IT!!!!
;           - The presence of "***" is important for marking the beginning of
;                 any data line.  The procedure "get_term" looks for this.
;           - It will handle any number of blank lines
;                                   a) before the "BATCH MODE" header,
;                                   b) between it and the first molecule,
;                                   c) between successive molecules to be processed
;                   -       It will handle one molecule with 3 or 4 fragments, for 3 properties.
;           Case 1 : Handle input of SPARC Data
;           Case 2 : When SPARC Data already exists, call to predicate checkfrag
;                       will fail.  This case makes note of why case 1 failed.
            ((getall_sparc ?name) if
                                                ;***** Write out Batch file for SPARC
                        (molecule ?name ?ee ? ?)
                        (molfrag ?ee ?ee1 ?ee2 ?ee3 ?ee4)
                        (not (checkfrag ?name ?ee))             ;Check to see if values already exist
                        (println "\nGetting values for the molecule " ?name)
                          (println "     with the Smiles string [" ?ee "].....")
                        (open "batch.dat" write sparc)
                        (sparcfrag_batch ?ee1)                  ;Output fragment data to make input
                        (sparcfrag_batch ?ee2)                  ;  batch file for SPARC
                        (sparcfrag_batch ?ee3)
                        (for (                          ; Special case where a fragment is a Hydrogen
                                        (not (= ?ee4 ""))
                                        (sparcfrag_batch ?ee4)      )
                             (
                                        (print "     Last fragment is a Hydrogen, using the ")
                                        (println "Smiles String [" ?ee4 "].")     )
                          )   ;*** End OR
                        (close sparc)
;                       (cut)
                                        ;***** Read in values generated by SPARC
                        (print "\nRun TO-SPARC.BAT now, followed by SPARC's Batch ")
                          (print "mode.  Hit return\n      when ready to proceed ")
                          (print "(i.e. TO SPARC).....")
                        (getline stdin ?) (println " ")
                        (open "batch.out" read sparckb)
                        (getline sparckb ?a)                            ; Prime pump
;                       (println "              First prime [" ?a "]")
                        (get_term ?a)                   ; Find "BATCH MODE OUTPUT" line
                        (getsparc_info ?pol1 ?alpha1 ?beta1 ?vol1)              ;Fragment 1 with ee1
;                         (print "     Frag1 val:[" ?pol1 "] [" ?alpha1 "] [" ?beta1 "]")
;                         (println " [" ?vol1 "].")
                        (getsparc_info ?pol2 ?alpha2 ?beta2 ?vol2)              ;Fragment 2 with ee2
;                         (print "     Frag2 val:[" ?pol2 "] [" ?alpha2 "] [" ?beta2 "]")
;                         (println " [" ?vol2 "].")
                        (getsparc_info ?pol3 ?alpha3 ?beta3 ?vol3)              ;Fragment 3 with ee3
;                         (print "     Frag3 val:[" ?pol3 "] [" ?alpha3 "] [" ?beta3 "]")
;                         (println " [" ?vol3 "].")
                        (or (                           ; Special case where a fragment is a Hydrogen
                                        (not (= ?ee4 ""))
                                        (getsparc_info ?pol4 ?alpha4 ?beta4 ?vol4)              ;Frag 4 with ee4
                                        (print "     Frag4 val: [" ?pol4 "] [" ?alpha4 "] [")
;                                       (println ?beta4 "] [" ?vol4 "].\n")
                                    )
                          (
                                        (println "          Last fragment was H, NO SPARC Data...")
                                        (= ?pol4 0) (= ?alpha4 0) (= ?beta4 0) (= ?vol4 0)
                                        (print "     Frag4 H: [" ?pol4 "] [" ?alpha4 "] [")
;                                       (println ?beta4 "] [" ?vol4 "].")
                                    )
                          )   ;*** End OR
                        (close sparckb)                         ; Close Mol file
                                        ;***** Save values generated by SPARC dynamically & to file
                        (println "  Saving the molecule data for " ?name)
                          (println "     to the SPARC property fragment knowledge base... ")
                        (open "fv-pol.kb" append fragval_kb)
                          (savef_value ?ee1 pol ?pol1)          (savef_value ?ee2 pol ?pol2)
                          (savef_value ?ee3 pol ?pol3)          (savef_value ?ee4 pol ?pol4)
                         (close fragval_kb)
                        (open "fv-alpha.kb" append fragval_kb)
                          (savef_value ?ee1 alpha ?alpha1)      (savef_value ?ee2 alpha ?alpha2)
                          (savef_value ?ee3 alpha ?alpha3)      (savef_value ?ee4 alpha ?alpha4)
                         (close fragval_kb)
                        (open "fv-beta.kb" append fragval_kb)
                          (savef_value ?ee1 beta ?beta1)        (savef_value ?ee2 beta ?beta2)
                          (savef_value ?ee3 beta ?beta3)        (savef_value ?ee4 beta ?beta4)
                         (close fragval_kb)
                        (open "fv-vol.kb" append fragval_kb)
                          (savef_value ?ee1 vol ?vol1)          (savef_value ?ee2 vol ?vol2)
                          (savef_value ?ee3 vol ?vol3)          (savef_value ?ee4 vol ?vol4)
                         (close fragval_kb)
            )
            ((getall_sparc ?name) if
                        (println "\n    All Values exist for the molecule " ?name ".")
            )

; (checkfrag ?name ?ee) - Check to see if fragment values for all fragments
;           and all properties already exist..
            ((checkfrag ?name ?ee) if
                        (molfrag ?ee ?ee1 ?ee2 ?ee3 ?ee4)
                        (println "  Checking to see if the SPARC values already exist... ")
;                         (println "     Molecule " ?name ", with the")
;                         (println "     Smiles string [" ?ee "]")
                        (fg-valu ?ee1 pol ?)                    ;Make sure values exist for frag 1
                        (fg-valu ?ee1 alpha ?)  (fg-valu ?ee1 beta ?)  (fg-valu ?ee1 vol ?)
                        (fg-valu ?ee2 pol ?)                    ;Make sure values exist for frag 2
                        (fg-valu ?ee2 alpha ?)  (fg-valu ?ee2 beta ?)  (fg-valu ?ee2 vol ?)
                        (fg-valu ?ee3 pol ?)                    ;Make sure values exist for frag 3
                        (fg-valu ?ee3 alpha ?)  (fg-valu ?ee3 beta ?)  (fg-valu ?ee3 vol ?)
                        (fg-valu ?ee4 pol ?)                    ;Make sure values exist for frag 4
                        (fg-valu ?ee4 alpha ?)  (fg-valu ?ee4 beta ?)  (fg-valu ?ee4 vol ?)
            )


; (sparcfrag_batch ?ee) - Output SPARC Batch mode data file for the Smiles
;           string ?ee.  As of 5/92, only polarizability and H bonding alpha's
;           and beta's are used.
; NOTE - Assumes that stream 'sparc' is already open.
;                 - Also see 'getsparc_info' where the properties that were calculated
;                       by SPARC are read in from the batch mode output file.
            ((sparcfrag_batch ?ee) if
```

```
                    (println "     Smiles string (" ?ss ") being used...")
                    (writeln sparc "['/25'].\n'" ?ss "'.\npol.")          ;polarizability
                    (writeln sparc "['/25'].\n'" ?ss "'.\nalpha.")        ;H-bonding alpha
                    (writeln sparc "['/25'].\n'" ?ss "'.\nbeta.")         ;H-bonding beta
                    (writeln sparc "['/25'].\n'" ?ss "'.\nvol.")          ;Molecular Volume
          )

; (getsparc_info ?value) - Get the 4 calculated properties from the SPARC
;          batch mode output file [BATCH.OUT].
;          Case 1 : Obtain the calculated value for the 4 properties & return them.
;          Case 2 : An error has occurred.  Should never execute this predicate.
; NOTE - 'getsparc_info assumes that the order of the properties in SPARCs
;                    batch mode input file (and hence output file) is as noted below.
;           - 'getsparc_record' assumes that stream 'sparckb' is already open.
;           - See 'getall_sparc' for the format of SPARCs batch mode output file.
;           - Also see 'sparcfrm batch' where the properties to be calculated
;                    by SPARC are written out to the batch mode input file.
          ((getsparc_info ?vala ?valb ?valc ?vald) if
                    (getsparc_record ? ? ?vala)                   ;get value for polarizability
                    (getsparc_record ? ? ?valb)                   ;get value for H-bonding alpha
                    (getsparc_record ? ? ?valc)                   ;get value for H bonding beta
                    (getsparc_record ? ? ?vald)                   ;get value for molecular volume
          ;         (println "          Finished getting values...")
                    (cut)
          )
          ((getsparc_info ? ? ? ?) if
                    (println "          RRROR in getsparc_info - EOF reached too soon...")
                    (cut) (fail)
          )

; (getsparc_record ?ss ?prop ?value) - Read a record of information from
;          the file [BATCH.OUT], generated by SPARC.  Return the molecules
;          Smiles String ?ss, the property that was calculated ?prop, and the
;          calculyyated value ?value.  This predicate reads an entire record and
;          does NOT read ahead to the next record.
;          Case 1 : Read in the entire record and return its information.
;          Case 2 : An error has occured.  Should never execute this predicate.
; NOTE - Assumes that stream 'sparckb' is already open.
;           - See 'getall_sparc' for the format of SPARCs batch mode output file.
          ((getsparc_record ?ss ?prop ?value) if
                    (getline sparckb ?a)
                    (get_term ?a)                                 ; check for "***" first in line
                    (not (eos sparckb))
                    (clear_line sparckb ?ss)                      ; Smiles string
                    (clear_line sparckb ?prop)                    ; Property
                    (clear_line sparckb ?13)                      ; Solvent mix
                    (sread sparckb ?) (sread sparckb ?) (get_char sparckb ?)
                      (getline sparckb ?14)                       ; get Temperature (1 less term)
                    (sread sparckb ?) (sread sparckb ?) (sread sparckb ?)
                      (get_char sparckb ?) (sread sparckb ?) (get_char sparckb ?)
                      (getline sparckb ?15)                       ; get Observed val (problem getting ')
                    (clear_line sparckb ?valu)                    ; Calculated value
                    (:= ?value (atrtonum ?valu))                  ; Convert string to numeric data
          ;         (print "Smiles string (" ?ss ")  Property (" ?prop ")\n Solvent ")
          ;         (print "Mixture (" ?13 ")  Temperature (" ?14 ")\n Observed ")
          ;         (println "Value's (" ?15 ")  Calculated Value's (" ?value ")")
                    (cut)
          )
          ((getsparc_record ? ? ?) if
                    (println "          RRROR in getsparc_record.  Reached EOF too soon.")
                    (cut) (fail)
          )

; (savef_value ?ssf ?valname ?fval) - Write out the fragment SMILES string
;          ?ssf, the property name ?valname, and the property value for the
;          fragment.
;          Case 1 : Fragment value doesn't currently exist in runtime KB, so save it
;          Case 2 : Fragment value already exists in runtime KB, so do nothing
;          Case 3 : Error - The value exists but isn't the same.  Could be that
;                    you are trying to compare text with numeric data from case 2.
; NOTE - assumes stream name 'fragval_kb' is already opened.
          ((savef_value ?ssf ?valname ?fval) if
                    (not (fg-valu ?ssf ?valname ?)) ;Make sure value doesn't exist
                    (writeterm fragval_kb ?ssf)     ;Write fragment e str to kb
                    (writeln fragval_kb " " ?valname " " ?fval)
                    (assert ((fg-valu ?ssf ?valname ?fval)))  ;Add val to runtime fv KB
                    (print "     Saved fragment (" ?ssf "), property (" ?valname)
                              (println "), value (" ?fval ").")
                    (cut)
          )
          ((savef_value ?ssf ?valname ?fval) if   ;Make sure if value exists, its
                    (fg-valu ?ssf ?valname ?fval)           ; the same as the one in the kb
          ;         (print "      Fragment (" ?ssf "), property (" ?valname "), value")
          ;                   (println " (" ?fval ") exists.")
                    (cut)
          )
          ((savef_value ?ssf ?valname ?fval) if   ;Error condition - Value exists
                    (fg-valu ?ssf ?valname ?fval2)           ; but it isn't the same
                    (print "      * Fragment (")
                    (not (= ?fval ?fval2))
                    (println ?ssf "), property (" ?valname "), exists.")
                    (println "          * KB value=(" ?fval2 "), new valu (" ?fval ").")
                    (cut)
          )
; (sparc_batch) - Output SPARC Batch mode data file for all properties.
; NOTE - Only want Polarizability, Alpha and beta values as of 5/1/92
          ((sparc_batch ?name) if
                    (molecule ?name ?ss ? ?)
                    (println "Smiles string (" ?ss ") being used...")
                    (open "batch.dat" write sparc)
          ;         (writeln sparc "[[1],[water]].\n['/25'].\n'" ?ss "'.\nsol.")
          ;         (writeln sparc "['/25'].\n'" ?ss "'.\nvp.")
          ;         (writeln sparc "['/25'].\n'" ?ss "'.\nhv.")
          ;         (writeln sparc "[[1],[water]].\n['/25'].\n'" ?ss "'.\ndist.")
          ;         (writeln sparc "['/25'].\n'" ?ss "'.\nbp.")
                    (writeln sparc "['/25'].\n'" ?ss "'.\npol.")
                    (writeln sparc "['/25'].\n'" ?ss "'.\nvol.")
          ;         (writeln sparc "['/25'].\n'" ?ss "'.\nind.")
                    (writeln sparc "['/25'].\n'" ?ss "'.\nalpha.")
                    (writeln sparc "['/25'].\n'" ?ss "'.\nbeta.")
                    (close sparc)
          )

; (load_sparc) - Loads results of property estimation calculated using
;          the program SPARC.  User is prompted for the SPARC output file
;          name, and "ld_sparc" does the loading.
          ((load_sparc) if
                    (print "File spec: ")
                    (getline stdin ?f)
                    (println "\n")
                    (ld_sparc ?f)
          )

; (ld_sparc ?output) - Loads results of property estimation calculated using
;          the program SPARC from the file ?output.  Information from each
;          molecule is loaded by "getinfo", which recursively calls itself to
;          process all the data in the file.
; NOTE
;          - This procedure assumes a specific output format from SPARC, namely:
;                                                  *** BATCH MODE OUTPUT ***
;
;                    *** Processing Molecule Number 1
;                    *** Smiles String: Cl-c(ccc1)cc1
;                    *** Property Type: Solubility
;                    *** Solvent Mixture: water
;                    *** Temperture: 25
;                    *** Observed Value's: (/25)
;                    *** Calculated Value: -4.276654759
;
;                    *** Processing Molecule Number 2
;                    *** Smiles String: Cl-c(cccc1)cc1
;          * NOTE that the escape sequences placed in the SPARC output file MUST
;                    BE REMOVED PRIOR TO READING IT!!!!
;          - The presence of "***" is important for marking the beginning of
;                    any data line.  The procedure "get_term" looks for this.
;          - It will handle any number of blank lines
;                    a) before the "BATCH MODE" header,
;                    b) between it and the first molecule,
;                    c) between successive molecules to be processed
;                    and will handle any number of molecules.
          ((ld_sparc ?output) if
                    (open ?output read sparckb)
                    (getline sparckb ?a)                          ; Prime pump
          ;         (println "First prime (" ?a ")")
                    (get_term ?a)                                 ; Find "BATCH MODE OUTPUT" line
                    (getline sparckb ?b)                          ; Prime pump
          ;         (println "Second prime (" ?b ")")
                    (get_term ?b)                                 ; Find first Processing molecule line
                    (getinfo)
                    (close sparckb)                               ; Close Mol file
                    (cut)
          )

; (get_term ?a) - Gets rid of blank lines from the input stream, and
;          checks that first non-blank line contains data, ie "***".
;          Case 1:
;          Case 2: removing blank lines from the input file
;          Case 3: checking that the first term on data lines is "***".
          ((get_term ?) if
                    (eos sparckb)
                    (println "          EOF for stream sparckb reached in get_term....")
                    (cut)
          )
          ((get_term ?a) if
          ;         (println "          Getting term case 1...")
                    (not (eos sparckb))
                    (= ?a "*")
          ;         (println "     Null line found.  Getting another term...")
                    (getline sparckb ?b)                          ; Prime pump
                    (get_term ?b)
                    (cut)
          )
          ((get_term ?a) if
```

```
                ;               (println "          Getting term case 2...")
                                (not (eos sparckb))
                ;               (println "      Current sparc kb line is\n          (* ?a *).")
                                (> (strpos ?a "**** 0) 0)
                ;               (println "      Getting another term...")
                                (cut)
                )

; (getinfo) - Processes the information from each molecule found in
;             the output file produced by the "BATCH" mode of SPARC.  Recursively
;             calls itself to process info on multiple molecules within one file.
;             Case 1: Process each molecule's information.
;             Case 2: Note the successful reading of the data file.
; NOTE - Assumes that first line of data in SPARC file, namely the line
;                       stating "*** Processing Molecule Number X" has already been
;                       read.  It also does not read the blank line seperating queries.
                ((getinfo) if
                                (not (eos sparckb))
                                (clear_line sparckb ?11)                        ; Smiles string
                                (clear_line sparckb ?12)                        ; Property
                                (clear_line sparckb ?13)                        ; Solvent mix
                                (sread sparckb ?) (sread sparckb ?) (get_char sparckb ?)
                                  (getline sparckb ?14)                 ; get Temperature (1 less term)
                                (sread sparckb ?) (sread sparckb ?) (sread sparckb ?)
                                  (get_char sparckb ?) (sread sparckb ?) (get_char sparckb ?)
                                  (getline sparckb ?15)                 ; get Observed val (problem getting ')
                                (clear_line sparckb ?16)                ; Calculated val
                                (println "Smiles string {" ?11 "}\nProperty {" ?12 "}")
                                (println "Solvent Mixture {" ?13 "}\nTemperature [" ?14 "]")
                                (println "Observed Value's {" ?15 "}\nCalculated Value's [" ?16 "]")
                                (assert ((sparc ?11 ?12 ?13 ?14 ?15 ?16)) )     ; Assert data
                                (println "  Look for more data..... \n")
                                (getline sparckb ?a)                            ; prime pump
                                (get_term ?a)                           ; check for "***" first in line
                                (getinfo)
                )
                ((getinfo) if
                                (println "Finished getting information.")
                                (eos sparckb)
                )

; (clear_line stream name data)  - Reads a line of data from the stream
;             indicated.  The first 3 terms and one character are discarded, and
;             the rest of the line is returned as "data".
                ((clear_line ?stream ?11) if
                                (sread ?stream ?) (sread ?stream ?) (sread ?stream ?)
                                (get_char ?stream ?) (getline ?stream ?11)
                )

;   ***END OF assert
)


;                               **** File NEW.HC ****                   February 26, 1993
(assert

; (anew) - abolish all predicates in new.hc so you an try again
                ((anew) if
                                (abolish run_one_prop)
                                (abolish list_all_mols)                 (abolish cnt_mols)
                                (abolish do_retract)                    (abolish test_cols)
                                (abolish list_mols)                             (abolish test_col)
                                (abolish test_all_sugg)                 (abolish test_all_cols)
                                (abolish ld_all)                                (abolish test_all_fg)
                                (abolish anew)                                  (consult new.hc)
                )


; (run_one_prop) - Run CHIRULE with only one property at a time.
                ((run_one_prop) if
                                (load_mol_kb)                           (load_molfrag_kb)
(load_chiral_kb)
                                (print "\n=====================================")
                                                (println "-------------------------")
                                (println "*** Generating Polarizability Property information .....")
                                (abolish fq-valu)                               (loadkb fq-valu "fv-pol.kb" YYN AAN)
                                (abolish fvalu_range)       (not (gen_fvrange_kb pol 1 10))
                                (test_cols)

                                (print "\n=====================================")
                                                (println "-------------------------")
                                (println "*** Generating Alpha Property information .....")
                                (abolish fq-valu)                               (loadkb fq-valu "fv-alpha.kb" YYN AAN)
                                (abolish fvalu_range)       (not (gen_fvrange_kb alpha 1 10))
                                (test_cols)

                                (print "\n=====================================")
                                                (println "-------------------------")
                                (println "*** Generating Beta Property information . ....")
                                (abolish fq-valu)                               (loadkb fq-valu "fv-beta.kb" YYN AAN)
                                (abolish fvalu_range)       (not (gen_fvrange_kb beta 1 10))
                                (test_cols)

                                (print "\n=====================================")
                                                (println "-------------------------")
                                (println "*** Generating Molecular Volume Property information ...")
                                (abolish fq_valu)                       (loadkb fq-valu "fv_vol.kb" YYN AAN)
                                (abolish fvalu_range)       (not (gen_fvrange_kb vol 100 10))
                                (test_cols)
                )

; (test_cols) - Run the four columns used to test the CHIRULE system.
                ((test_cols) if
                 (print "\n=====================================")
                                (println "-------------------------")
                 (println "*** Testing all columns for predicate test_col .....")
                                (or             (test_col "Crownpak CR")
                                                (test_col "Chiralcel OR")
                                                (test_col "Chiralcel OD")
                                                (test_col "Chiralpak WH")
                                                (println)
                                )               ;end OR
                )

; (list_mols ?column) - List out and count all of the molecules that can be
;             separated using column ?column.
; NOTE - TESTING BY COLUMN
                ((list_mols ?col) if
                                (print "\n=========================================")
                                                (println "-----------------------")
                                (println "*** Looking for molecules with column " ?col " ....")
                                (assert ((molcount 0)))
                                (cnt_mols ?col)
                                (retract ((molcount  ?num)))
                                (print "*** Found a total of " ?num " molecules for column " ?col)
                                (print ".\n=========================================")
                                                (println "---------------\n")
                                (fail)
                )

; (test_col ?column) - Find a molecule that can be separated using column
;             ?column and see what column is suggested by CHIRULE.
; NOTE - TESTING BY COLUMN
                ((test_col ?column) if
                                (chiral_col ?molss ?column ?rel)
                                (print "\n=========================================")
                                                (println "-----------------------")
                 (print "*** Testing for suggestions for column " ?column " ....")
                                (molecule ?name ?molss ? ?)
                ;               (println "    Found " ?name)
                                (retcr ?name)
                                (fail)
                )

; (test_all_cols ?pred) - Call either 'list_mols ?col' or 'test_col ?col' to
;             list and count OR ask for suggestions for all molecules that can be
;             separated by the particular columns listed below.
; NOTE - TESTING BY COLUMN
                ((test_all_cols ?pred) if
                 (print "\n=====================================")
                                (println "-------------------------")
                 (println "*** Testing all columns for predicate " ?pred " .....")
                                (or             (?pred "Crownpak CR")
                                                (?pred "Chiralcel OR")
                                                (?pred "Chiralcel OD")
                                                (?pred "Chiralpak WH")
                                                (?pred "Chiralcel CA-1")
                                ;               (?pred "Chiralcel OA")
                                                (?pred "Chiralcel OJ")
                                                (?pred "Chiralcel OK")
                                                (?pred "Chiralcel OC")
                                                (?pred "Chiralcel OF")
                                                (?pred "Chiralcel OG")
                                                (?pred "Chiralpak WM")
                                                (?pred "Chiralpak WR")
                                                (?pred "Chiralpak OP")
                                                (?pred "Chiralpak OT")
                                                (println)
                                )               ;end OR
                )

; (list_all_mols) - List out and count all molecules within the molecule
;                   knowledge-base.
; NOTE - TESTING BY MOLECULE
                ((list_all_mols) if
                                (print "\n=========================================")
                                                (println "-----------------------")
                                (println "*** Listing all molecules in the MOL Knowledge base .....")
                                (assert ((molcount 0)))
                                (cnt_mols ?)
                                (retract ((molcount  ?num)))
                                (println "*** Found a total of " ?num " molecules.")
                                (print "=========================================")
                                                (println "--------------------\n")
                                (cut)
```

```
        )

; (cnt_mole ?col) - Go through the entire molecule knowledge base and count
;       the number of molecules.  If ?col contains a column name, only molecules
;       that can be separated with that column will be counted.  If ?col is not
;       specified, all molecules will be counted.  Before it is called,
;       "molcount 0" must be asserted to initialize the count.  After it is
;       called "molcount ?num" must be retracted, and ?num will contain the
;       number of molecules found.
;       Case 1: Accumulate count, use fail to count ALL molecules
;       Case 2: Need to prevent further backtracking.
        ((cnt_mole ?col) if
                (molecule ?name ?molss ? ?)
                (do_retract ?molss ?col ?old)
                (:= ?new (+ ?old 1))
                (assert ((molcount ?new)))
                (println "      Found Molcule (" ?new ") " ?name)
                (fail)
        )
        ((cnt_mole ?) if
                (cut)           ;(println " **** Done counting ****")
        )

; (do_retract ?molss ?col ?old) - This predicate is necessary so that choice
;       points are not left either at the retract or the selection of a column.
;       In "cnt_mole" if "?col" isn't specified, this makes sure each molecule
;       is only counted once even though each molecule may have several column
;       references.  If "?col" is specified, it will fail if the molecule chosen
;       does not have column "?col" as a column reference, hence only counting
;       molecules that have "?col" as a reference.
        ((do_retract ?molss ?col ?old) if
                (chiral_col ?molss ?col ?)
                (retract ((molcount ?old)))                             (cut)
        )

; (test_all_sugg) - call 'tester ?mol' to ask for a column suggestion for
;       each of the molecules listed below
; NOTE - TESTING BY MOLECULE
        ((test_all_sugg) if
                ;       (tester "PHENYLGLYCINE")                (println "")  ;Testing CR
                ;       (tester "1-PHENYLETHYLAMINE")           (println "")
                ;       (tester "L-ISOLEUCINE")                 (println "")

                ;       (tester "1-PHENYLETHANOL")              (println "")  ;Testing OR
                ;       (tester "N,O-DIBENZOYL2-AMINOBUTANOL")  (println "")
                ;       (tester "N-(PHENYLETHYL)ACETAMIDE")     (println "")

                ;       (tester "ALPRENOLOL")                   (println "")  ;Testing OO
                ;       (tester "CARTEOLOL")                    (println "")
                ;       (tester "1-(1-HYDROXYETHYL)NAPHTHALENE")(println "")

                ;       (tester "DL-VALINE")                    (println "")  ;Testing NH
                ;       (tester "N-ACETYL TRYPTOPHAN")          (println "")
                        (tester "L-METHIONINE")                 (println "")

                        (tester "MANDELAMIDE")                  (println "")  ;Testing CA 1
                        (tester "METHYL MANDELATE")             (println "")  ;Testing CR,OR
                        (tester "METHYL2 PHENYLPROPIONATE")     (println "")  ;Testing OO
        )

; (ld_all) - load all MOL files
        ((ld_all) if
                        (ld_mol t1.mol ?)           ;PYRIDOXAL      (OH, C=O, PYR)
                        (ld_mol ts1.mol ?)          ;TS1            (COOH, C=C)
                        (ld_mol t2.mol ?)           ;ETHYL BENZOATE                 (BENZ, COOX)
                        (ld_mol t3.mol ?)           ;PHTHALIC ANHYDRIDE             (R2O, R5)
                        (ld_mol t4.mol ?)           ;PYRIDOXAL 5-PHOSPHATE          (PO3, R2O, PYR)
                        (ld_mol t5.mol ?)           ;Testing-fg     (-CC-, NH2, COOH)
                        (ld_mol t5a.mol ?)          ;Tet-H          ( CC , NH2 [H's specified], COOH)
                        (ld_mol t6.mol ?)           ;T6             ( NH2,  NHR,  NR2)
                        (ld_mol tripl.mol ?)        ;TRIPPLE-CC     (-CC-, NO2, PO2H2)
                        (ld_mol tripl-k.mol ?)      ;TRIPPLE-KETN                   (-CC-, X2CO, NO2, PO2H2)
                        (ld_mol tripl-n.mol ?)      ;TRIPPLE-CN     ( CN, NO2, PO2H2)
        )

; (test_all_fg) - Go through tests for all functional group predicates
        ((test_all_fg) if
                        (println "\n\n\n++++++++++ Testing (CdC ?) ++++++++++ ")
                        (not (CdC ?))
                        (println "\n\n\n++++++++++ Testing (C#C ?) ++++++++++ ")
                        (not (C#C ?))
                        (println "\n\n\n++++++++++ Testing (C#N ?) ++++++++++ ")
                        (not (C#N ?))
                        (println "\n\n\n++++++++++ Testing (cmth ?) ++++++++++ ")
                        (not (cmth ?))
                        (println "\n\n\n++++++++++ Testing (OmH ?) ++++++++++ ")
                        (not (OmH ?))
                        (println "\n\n\n++++++++++ Testing (CdO ?) ++++++++++ ")
                        (not (CdO ?))
                        (println "\n\n\n++++++++++ Testing (CHdO ?) ++++++++++ ")
                        (not (CHdO ?))
                        (println "\n\n\n++++++++++ Testing (X2CdO ?) ++++++++++ ")
                        (not (X2CdO ?))
                        (println "\n\n\n++++++++++ Testing (COOH ?) ++++++++++ ")
                        (not (COOH ?))
                        (println "\n\n\n++++++++++ Testing (COOX ?) ++++++++++ ")
                        (not (COOX ?))
                        (println "\n\n\n++++++++++ Testing (Chex ?) ++++++++++ ")
                        (not (Chex ?))
                        (println "\n\n\n++++++++++ Testing (Benz ?) ++++++++++ ")
                        (not (Benz ?))
                        (println "\n\n\n++++++++++ Testing (Pyrdn ?) ++++++++++ ")
                        (not (Pyrdn ?))
;                       (println "\n\n\n++++++++++ Testing ( ?) ++++++++++ ")
;                       (not ( ?))
        )

;   ***END OF assert
)

;                       **** File CHIRULE.HC ****               February 24, 1993
(assert

; (achirule) - abolish all predicates in chirule.hc so you can try again
        ((achirule) if
                        (abolish diff_offset)                   (abolish p_norm)
                        (abolish a_norm)                                (abolish b_norm)
                        (abolish v_norm)
                        (abolish fragmatch)                     ;Runtime fragment match kb
                        (abolish column-sugg)       ;Runtime column suggestion kb
                        (abolish add2chiral)                    (abolish show_chiral)
                        (abolish chirule)
                        (abolish tester)                                (abolish tester)
                        (abolish msimilar)                              (abolish fsimilar)
                        (abolish psimilar)
                        (abolish calc_diff)                             (abolish assert_colsugg)
                        (abolish column_value)                  (abolish get_property)
                        (abolish make_col_list)                 (abolish order_col_list)
                        (abolish find$emal1$et)                 (abolish print_col_list)
                        (abolish remove$dups)
                        (abolish rpt_chiral_info)  (abolish getinfo_chiral)
                        (abolish achirule)                              (consult chirule.hc)
        )

        ((diff_offset 1000)) ;Arbitrary scaling factor for difference calculations
        ((p_norm 1))                    ;Normalizing factors, based on the range of values
        ((a_norm 2))                    ;       for each property found in its knowledge base
        ((b_norm 2))                    ;       fragment value file FV-xxxxx.KB
        ((v_norm 200))

; (add2chiral) - Add a new molecule to the Chiral Expert System knowledge base.
;       Add a molecule to the runtime molecule KB, find its Smiles string,
;       save the molecule in the static molecule KB, fragment the molecule
;       around the chiral center and save them to the mol-frag KB,
;       calculate the properties for each fragment and save them to the
;       mol-valu KB, and finally get the column and reference and add it
;       to the chiral KB.
        ((add2chiral) if
                        (print " \nEnter the file spec (including the extension) for the ")
                                        (print "Molecule\n      to add to the Chiral System: -> ")
                        (getline stdin ?f)              ;get the file name with extension
                        (ld_mol ?f ?name)               ;load the molecule into runtime mol KB
                        (get_smiles ?name)              ;get the smiles string for runtime mol KB
                        (getall_smiles ?name)       ;ck4 chrl, fgmt_mol, frag SS, add2 frag KB
                        (getall_sparc ?name)        ;get 3 properties 4each frag, add2 valu KB
                        (getinfo_chiral ?name)      ;get column & reference, add2 chiral KB
                        (save_mol ?name)                ;save the molecule in the static mol KB
                        (saveall_smiles ?name)      ;save fragment info to static mol-frag KB
                        (cut)
        )

; (show_chiral ?name) - Display all of the current information for the
;       molecule ?name.  If ?name isn't supplied, it will display all
;       information for each molecule in the molecule knowledge base.
        ((show_chiral ?name) if
                        (molecule ?name ?ss ? ?)
                        (println "{molecule " ?name " " ?ss "....}\n")
                        (molfrag ?ss ?ssf1 ?ssf2 ?ssf3 ?ssf4)
                        (print "{molfrag " ?ss "   " ?ssf1 "      " ?ssf2 "     " ?ssf3 "     "
                                        (println ?ssf4 "}\n")
                        (fg-valu ?ssf1 pol ?pol1)               (fg-valu ?ssf2 pol ?pol2)
                        (fg-valu ?ssf3 pol ?pol3)               (fg-valu ?ssf4 pol ?pol4)
                        (print "{fg-valu " ?ss " pol " ?pol1 " " ?pol2 " " ?pol3 " "
                                        (println ?pol4 "}\n")
                        (fg-valu ?ssf1 alpha ?alpha1)           (fg-valu ?ssf2 alpha ?alpha2)
                        (fg-valu ?ssf3 alpha ?alpha3)           (fg-valu ?ssf4 alpha ?alpha4)
                        (print "{fg-valu " ?ss " alpha " ?alpha1 " " ?alpha2 " " ?alpha3 " "
                                        (println ?alpha4 "}\n")
                        (fg-valu ?ssf1 beta ?beta1)             (fg-valu ?ssf2 beta ?beta2)
                        (fg-valu ?ssf3 beta ?beta3)             (fg-valu ?ssf4 beta ?beta4)
                        (print "{fg-valu " ?ss " beta " ?beta1 " " ?beta2 " " ?beta3 " "
                                        (println ?beta4 "}\n")
                        (fg-valu ?ssf1 vol ?vol1)               (fg-valu ?ssf2 vol ?vol2)
```

```
                    (fq-valu ?ss1 vol ?vol3)                (fq-valu ?ss4 vol ?vol4)
                    (printf "(fq-valu " ?ss " vol " ?vol1 " " ?vol2 " " ?vol3 " "
                            (println ?vol4 ")\n")
                    (chiral col ?ss ?column ?ref)
                    (println "(chiral " ?ss " [* ?column *] [* ?ref *])\n\n\n")
          )

; (chirule)  Run the chiral column suggestion system CHIRULE.  It will
;            suggest a chiral column for a
;            newly entered molecule of interest.  Enter the name of the molecule,
;            load it tothe runtime molecule KB, make sure that it has a chiral
;            atom, find it's Smiles string, fragment the molecule around the chiral
;            center and save them to the mol-frag KB, calculate the properties
;            for each fragment and save them to the mol-valu KB, and finally
;            search the chiral KB to find a suggested column
; NOTE - ld_mol, get_smiles, getall_smiles, and getall_sparc all
;            check to see if there respective data already exists for the
;            current molecule.
          ((chirule) if
                    (println "\n      ***** Welcome to CHIRULE!! *****\n")
                    (print "Enter the file spec (including the extension) for the *)
                           (print "Chiral Molecule\n     you wish to determine a *)
                           (print "Chiral column:  > *)
                    (getline stdin ?f)                      ;get the file name with extension
                    (ld_mol ?f ?name)                       ;load the NEW molecule
                              ;  Checks name only
                              ;  "check_molname ?" will add or replace RUNTIME version ONLY...
                    (molecule ?name ? ?atlist ?)            ;Get ?atlist for chiral test
                    (find_chiral ?atlist ?cnum)             ;Make sure chiral atom exists
                    (not (= ?cnum 0))                                ;  Returns "0" if none found
;                   (println "     Found chiral atom at atom number " ?cnum ".")
                    (get_smiles ?name)                      ;get the smiles string for NEW molecule
                              ;  Checks name and smiles string
                              ;  "check_ss ? ?" will add or replace RUNTIME version of
                              ;   "molecule" KB ONLY...
                    (getall_smiles ?name)       ;fqmt NEW mol, find frag SS, add2 frag KB
                              ;  Makes sure that a chiral carbon and the Smiles String
                              ;  exists, and that there that there is a chiral atom.
                              ;  Will ask if you want to add to the runtime fragment KB.
                              ;  "check_molfrag ? ?" will allow addition or replacement
                              ;  of RUNTIME version ONLY...  BUT it does save "molfrag" KB
                              ;  to file "MOL-FRAG.KB".
                    (getall_sparc ?name)        ;get 3 properties 4each frag, add2 valu KB
                              ;  Makes sure values don't already exist in the fq valu KB.
                              ;  If they do, it won't currently allow you to change them
                              ;  in either the runtime or static KB.
                              ;  Makes changes to "fq-valu" KB and saves new fragment info
                              ;  to "FV-xxxx.KB" for each property.
                              (abolish fvalu-range)
                              (gen_all_range 10)
                    (print "\n*** Looking for Molecules Similar to " ?name ".")
                    (molecule ?name ?ss1 ? ?)  ;get Smiles String for similarity srch
                    (msimilar ?ss1 ?ss2 ?del)  (cut)          (print ".")
                    (calc_diff)          (cut)                                          (print ".")
                    (make_col_list ?col-list) (cut)                          (print ".")
                    (order_col_list ?col-list ?ocol-list) (cut)
                    (print "\n - Ordered suggestions (column) for ")
                      (println ?name "....\n")
                    (print_col_list ?ocol-list)          ;???
                    (println "\nDo you want to add the molecule " ?name )
                      (print "     to the static MOLECULE, SMILES, and COLUMN *)
                      (print "knowledge base?\n     [Y-yes, N-no (default)]  Y/N -> *)
                    (read ?Char) (getline stdin ?)
                    (:= ?ans [strcase ?Char upper])          (cut)
          (or (     (= "Y" ?ans)
                              (save_mol ?name)              ;save NEW mol to the static mol KB
                              (saveall_smiles ?name)        ;save NEW mol to static mol-frag KB
                    (cut) (getinfo_chiral ?name)            ;save NEW mol to static column KB
                    )
              (println "\nDone running CHIRULE.\n")
            )               ;END OR
          )

; (tester)  Input an existing molecule name to test the CHIRULE system,
;           to find out if there are any similar molecules in the Molecule KB.
          ((tester) if
                    (print "     Molecule name -> *)
                    (getline stdin ?mol)
                    (tster ?mol)
          )

; (tster ?mol) - Test the CHIRULE system using the existing molecule name ?mol,
;           testing if there are any molecules similar to ?mol in the molecule KB.
          ((tster ?mol) if
                    (molecule ?mol ?ss1 ? ?)
                    (print "\n - Ordered column suggestions [column] for molecule " ?mol)
;                   (get_delta ?delta) to progress from highest to lowest delta values
                    (msimilar ?ss1 ?ss2 ?)      (cut)         (print ".")   ;Last ARG ?delta
                    (calc_diff)                               (cut)         (print ".")
                    (make_col_list ?col-list) (cut)          (println "...")
;                   (println "***  Column list *** \n " ?col-list "\n ***********")
                    (order_col_list ?col-list ?ocol-list) (cut)
                    ;             (println "***  Ordered Column list *** \n " ?ocol-list "\n *******")
                    (print_col_list ?ocol list)
          )

; (msimilar ?ssa ?ssb ?delta) - Search molecule fragment knowledge base for a
;           smiles string ?ssb that is similar to the provided smiles string ?ssa.
;           If ?delta is provided, use it.  If ?delta isn't provided, the value
;           used will be reported back.
; NOTE - Predicate will NOT allow for testing for similarity of identical
;           molecules (see statement testing for inequality of smiles string)
          ((msimilar ?ssa ?ssb ?) if
                    (molfrag ?ssa ?fa1 ?fa2 ?fa3 ?fa4)                    ; Known smiles string
                    (molfrag ?ssb ?fb1 ?fb2 ?fb3 ?fb4)                    ; Smiles string to test
                    (not (= ?ssa ?ssb))                  ; Make sure they aren't identicle
                    (removedups (?fa1 ?fa2 ?fa3 ?fa4) (?fb1 ?fb2 ?fb3 ?fb4) ?fla ?flb)
                    (molecule ?mol2 ?ssb ? ?)
                    (listsiz ?fla ?nmatch)
;                   (println " * Molecule (* ?mol2 *], F lists: " ?fla "," ?flb ".")
                    (fsimilar ?ssa ?ssb ?nmatch ?fla ?flb ? ?)
                    (fail)
          )
          ((msimilar ? ? ?) if
;                   (println " - Finished with runtime column match KB (fragmatch)...")
                    (cut)         )

; (fsimilar ?ssa ?ssb ?nmatch ?fla ?flb ?property ?delta) - Demonstrate that
;           each fragment in list ?fla has a corresponding fragment that
;           can shown to have similar property values with a member in the fragment
;           list ?flb.
;           Case 1 : Base Case - No more fragments to look at.
;           Case 2 : See if fragment ?h from first list matches any of the
;                    fragments in the second list.  If not, the predicate will fail,
;                    and msimilar will go on to another molecule.
          ((fsimilar ?ssa ?ssb ?nmatch nil nil ? ?) if
;                   (println "          End of (list.")
                    (cut) )
          ((fsimilar ?ssa ?ssb ?nmatch (?h.?t) ?fl1 ?property ?delta) if
;                   (println "          Looking for " ?h " in list " ?fl1 "...")
                    (psimilar ?ssa ?ssb ?nmatch ?h ?fl1 ?fl2 ?property ?delta)
;                   (println "          Found " ?h ", leaving list " ?fl2 ".")
                    (fsimilar ?ssa ?ssb ?nmatch ?t ?fl2 ?property ?delta)
          )

; (psimilar ?ssa ?ssb ?nmatch ?h ?fl1 ?fl2 ?property ?delta) - Demonstrate that
;           for the property ?property there exists a match between the fragment ?h
;           and one of the fragments in the fragment list ?fl1 and assert info into
;           the "fragmatch" runtime knowledge base.
;           NOTE - that if no match is found for ?h in the fragment list ?fl1, the
;                  predicate will fail, hence fsimilar will fail, and msimilar will go
;                  on to another molecule.
;                - the "not" below in case 1 eliminates those cases whose fvalu-range
;                  low and high values are both the same.
;           Case 1 : Found that the two fragments ?h and ?h2 match
;           Case 2 : First fragment in fragment list ?fl1 didn't match with ?h, so
;                    add it to the answer list ?fl2 and look at rest of list ?fl1.
          ((psimilar ?ssa ?ssb ?nmatch ?h (?h2.?t2) ?f2 ?prop ?delta) if
                    (fvalu-range ?h ?prop ?delta ?y)
                    (fvalu-range ?h2 ?prop ?delta ?y)
                    (not (fragmatch ?ssa ?ssb ?nmatch ?h ?h2 ?prop ?delta ?))
                    (fq-valu ?h ?prop ?diff1)
                    (fq-valu ?h2 ?prop ?diff2)
                    (diff_offset ?offset)
                    (:= ?diff [/ [* ?offset [abs [- ?diff1 ?diff2]]] ?delta ])
;                   (print "          (fragmatch " ?h ", " ?h2 ", " ?nmatch ", " ?prop)
;                          (println ", value " ?y ", delta " ?delta "), list " ?t2 ".")
                    (assert ((fragmatch ?ssa ?ssb ?nmatch ?h ?h2 ?prop ?delta ?diff)) )
          )
          ((psimilar ?ssa ?ssb ?nmatch ?h (?h2.?t2) (?h2.?t3) ?property ?delta) if
;                   (println "          No Match of " ?h " & " ?h2 ", list " ?t2 ".")
                    (psimilar ?ssa ?ssb ?nmatch ?h ?t2 ?t3 ?property ?delta)
          )

; (calc_diff) - Calculates the metric indicating how closely the molecule
;           of interest with smiles string ?ssa is to the knowledge base molecule ?ssb.
;           Note that as long as a molecule has been found to be similar to
;           another for at least one property, the total difference will be
;           calculated.  This is because calc_diff picks out the fragment match
;           regardless of which property it is, and passes the matching molecule
;           and matching fragments on to column_value to calculate the difference
;           for each property.  Case 1 - 4 all fail so that all of the entries in
;           the fragmatch knowledge base are utilized.
;           Case 1 : Only 1 fragment not identical, so calculate value
;           Case 2 : 2 fragments not identical.  Calculate the value for the
;                    current fragments.  Predicate "assert_column" will handle how
;                    to correctly assert data to column kb.  Note that column_value
;                    will retract "fragmatch" for these fragments.
;           Case 3 : Only 1 fragment identical - Not implemented.
;           Case 4 : No fragments identical - Not implemented.
;           Case 5 : Cases 1 & 2 will continue on as long as there is data in the
;                    fragmatch knowledge base.  When other two have failed, you are done.
          ((calc_diff) if
                    (fragmatch ?ssa ?ssb 1 ?frag ?frag2 ? ? ?)
```

```
;               (molecule ?mol ?smb ? ?)
;               (println " + Looking for simple fragment for molecule (" ?mol ")...")
                (column_value ?smx ?smb 1 ?frag ?frag2 0 ?value ?)
                (assert_colmugg ?smx ?smb ?value)                       ;Assert data
;               (println " + Found simple fragment with a matching value " ?value ".")
                (fail)
)
((calc_diff) if
                (fragmatch ?smx ?smb 2 ?fraga ?frag1 ? ? ?)             ;Get 1st pair of frags
;               (molecule ?mol ?smb ? ?)
;               (println " + Looking for double fragment for molecule (" ?mol ")...")
                (fragmatch ?smx ?smb 2 ?fragb ?frag2 ? ? ?)             ;Get 2ndpair frags
                (not (= ?fraga ?fragb))     (not (= ?frag1 ?frag2))     ;Check not same
 (column_value ?smx ?smb 2 ?fraga ?frag1 0 ? ?tot1 )   ;Get diff value
 (column_value ?smx ?smb 2 ?fragb ?frag2 ?tot1 ? ?tot)                  ;Get 2nd diff value
                (assert_colmugg ?smx ?smb ?tot)                         ;Assert data
;               (println " + Found double fragment with a matching value " ?tot ".")
                (fail)
)
((calc_diff) if
                (fragmatch ?smx ?smb 3 ?fraga ?frag1 ? ? ?)             ;Get 1st pair of frags
;               (molecule ?mol ?smb ? ?)
;               (println " + Looking for tripple frag for molecule [" ?mol "]...")
                (fragmatch ?smx ?smb 3 ?fragb ?frag2 ? ? ?)             ;Get 2ndpair frags
                (not (= ?fraga ?fragb))     (not (= ?frag1 ?frag2))     ;Check not same
                (fragmatch ?smx ?smb 3 ?fragc ?frag3 ? ? ?)             ;Get 3rdpair frags
                (not (member ?fragc (?fraga ?fragb)))                   ;Check not same
                (not (member ?frag3 (?frag1 ?frag2)))
 (column_value ?smx ?smb 3 ?fraga ?frag1 0 ? ?tot1)    ;Get diff value
 (column_value ?smx ?smb 3 ?fragb ?frag2 ?tot1 ? ?tot2);Get 2nd diff value
 (column_value ?smx ?smb 3 ?fragc ?frag3 ?tot2 ? ?tot)                  ;Get 3rd diff value
                (assert_colmugg ?smx ?smb ?tot)                         ;Assert data
;               (println " + Found tripple frag with a matching value " ?tot ".")
;               (print "     Fragments: " ?fraga " & " ?frag1 ", " ?fragb " & ")
;                 (println ?frag2 ", " ?fragc " & " ?frag3 ".")
                (fail)
)
((calc_diff) if
                (fragmatch ?smx ?smb 4 ?fraga ?frag1 ? ? ?)
                (molecule ?molb ?smb ? ?)
                (print "*** 4 different fragments " ?molb)
                  (println ".  Not yet implimented...")
                (fail)
)
((calc_diff) if
                (abolish fragmatch)
;               (print "\n +     Finished calculating differences, ")
;                 (println "abolishing fragmatch....")
)

; (column_value ?smx ?smb ?nmatch ?frag ?frag2 ?oldtot ?value ?newtot) -
;             Combine the "likeness" metric from each of the properties, normalizing
;             each of them, then adding them together.  ?oldtot passes in the total
;             from previous fragment calculations, this is added to ?value which is
;             the total for all properties from these 2 fragments.  If this new total
;             ?newtot is greater than the cutoff (10,000 here), it fails and isn't
;             asserted into the column suggestion knowledge base by predicate
;             "calc_diff".
              ((column_value ?smx ?smb ?nmatch ?frag ?frag2 ?oldtot ?value ?newtot) if
                (get_property ?smx ?smb ?nmatch ?frag ?frag2 pol ?p_diff)
                (get_property ?smx ?smb ?nmatch ?frag ?frag2 alpha ?a_diff)
                (get_property ?smx ?smb ?nmatch ?frag ?frag2 beta ?b_diff)
                (get_property ?smx ?smb ?nmatch ?frag ?frag2 vol ?v_diff)
                (p_norm ?p_nlz)             (:= ?p_val (/ ?p_diff ?p_nlz))
                (a_norm ?a_nlz)             (:= ?a_val (/ ?a_diff ?a_nlz))
                (b_norm ?b_nlz)             (:= ?b_val (/ ?b_diff ?b_nlz))
                (v_norm ?v_nlz)             (:= ?v_val (/ ?v_diff ?v_nlz))
                (:= ?value (+ ?v_val (+ ?p_val (+ ?a_val ?b_val))))
                (:= ?newtot (+ ?oldtot ?value))
                (> 1000 ?newtot)
;               (println " *          ev final value -> " ?value ".")
                (cut)
              )

; (assert_colmugg ?smx ?smb ?value) - Handles the correct way to go about
;             asserting data to the "column" knowledge base when there are 2 fragments that
;             are found not to be identical but to be similar.
;             Case 1 : Entry not found in the column knowledge base, so add it.
;             Case 2 : Entry found, but current entry is less than newly calculated
;                      value, so drop new value.
;             Case 3 : Entry found and newly calculated value is lower, so retract
;                      current value in column kb and replace it.
              ((assert_colmugg ?smx ?smb ?value) if    ;Entry not found
                (not (column-mugg ?smx ?smb ?))
;               (println "      Found entry not in column kb, add " ?value ".")
                (assert ((column-mugg ?smx ?smb ?value)))               )
                (cut)
              )
              ((assert_colmugg ?smx ?smb ?values) if   ;Found, but current is lower
                (column-mugg ?smx ?smb ?valueb)          ; so do nothing
                (<= ?valueb ?values)
;               (println "       Found " ?valueb " less than " ?values ", so OK.")
                (cut)
              )
              ((assert_colmugg ?smx ?smb ?values) if   ;Found, and new value is lower
                (column-mugg ?smx ?smb ?valueb)
                (< ?valueb ?values)                      ; not needed, added for completeness
;               (println "       Found " ?valueb " < than " ?values ", swap.")
                (cut)
                (retract ((column-mugg ?smx ?smb ?valueb))              )  ;retract old ?valueb
                (assert ((column-mugg ?smx ?smb ?values))               )  ;assert new ?values
              )

; (get_property ?smx ?smb ?nmatch ?frag ?frag2 ?property ?diff) - Return
;             the difference between the two fragments ?frag and ?frag2.  If
;             Case 1 : Value exists in fragmatch kb, so retract it an return ?diff
;             Case 2 : Value doesn't exist in the fragmatch kb, so calculate it.
;                      Since wasn't found in fragmatch, assume delta=1.
;             Case 3 : Property doesn't exist in Fragment Value Kb, so diff is zero.
;                      Need when testing without loading one of the properties.
;             NOTE - Don't need delta values.  fragmatch kb already takes into
;                    account delta values in its ?diff values (see psimilar).
              ((get_property ?smx ?smb ?nmatch ?frag ?frag2 ?property ?diff) if
                (fragmatch ?smx ?smb ?nmatch ?frag ?frag2 ?property ?delta ?diff)
;               (print "Retracting " ?property " (fragmatch ?frag " " ?frag2)
;                 (println " " ?diff ").")
                (cut)
              )
              ((get_property ?smx ?smb ?nmatch ?frag ?frag2 ?property ?diff) if
                (fg-valu ?frag ?property ?diff1)
                (fg-valu ?frag2 ?property ?diff2)
                (diff_offset ?offset)
                (:= ?diff (+ ?offset (abs [- ?diff1 ?diff2]))))         ;Assume delta=1
;               (print "Uncalculated " ?property " difference "?frag " ")
;                 (println ?frag2 " " ?diff ").")
                (cut)
              )
              ((get_property ? ? ? ? ? ? 0) if (cut))

; (make_col_list ?col-list) - Make the column suggestion list by retracting
;             the data from the runtime knowledge base "column-mugg".  The list is built
;             as the predicate recurses out in ?col-list.  The information contained
;             in each member of the list is the same as in the knowledge base "column-mugg",
;             namely the &miles String of the Molecule of interest, the SS of the
;             matching molecule, and the match value.
;             Case 1 : Entries in the knowledge base "column-mugg" exist, so retract one
;                      and recursively call itself.
;             Case  2 : When Case 1 fails, there are no more entries, so stop.
; NOTE: Could put other info in the list, like name of matching molecule.
              ((make_col_list ((?smx ?smb ?values).?col-list)) if
                (retract ((column-mugg ?smx ?smb ?values))              )
                ;Other info for list would be added here.
                (cut)           (make_col_list ?col_list)
              )
              ((make_col_list nil) if
                (not (column-mugg ? ? ?))
;               (println " + Finished making column suggestion list...\n")
              )

; (order_col_list ?col-list ?ordered-list) - Order the list of column
;             suggestions ?col-list according to the value of the match, lowest
;             match first, and return it in ?ordered-list.
;             See "make_col_list" for the format of the data for column suggestions.
;             Case 1 : No more column suggestions, so finished.
;             Case 2 : Remove the first suggestion, use it to prime pump, and
;                      call "findsmallest", which will return the suggestion with
;                      the smallest match value.  Add this to the front of the
;                      ordered suggestion list ?ordered-list, and recursively call
;                      "findsmallest" with the suggestions remaining, ?latb.
              ((order_col_list nil nil) if            (cut)
;               (println " + Finished ordering column suggestions.....\n")
              )
              ((order_col_list ((?sa ?sb ?va).?lsta) ((?s1 ?s2 ?v1).?lst1) ) if
                (findsmallest (?sa ?sb ?va) ?lsta (?s1 ?s2 ?v1) ?lstb)
                (cut) (order_col_list ?lstb ?lst1)
              )

; (findsmallest ?test ?in-list ?smallest ?out-list) - From the column
;             suggestion list ?in-list, find the suggested column with the smallest
;             match value, ?smallest, and return the rest of the column suggestions
;             in ?out_list.  ?test is used to prime the pump for the procedure.
;             See "make_col_list" for the format of the data for column suggestions.
;             Case 1 : There are no more suggestions in the list.  Build the
;                      ordered column suggestion list as the predicate recurses out.
;             Case 2 : The match value ?c of the proposed column suggestion (?a ?b ?c)
;                      is smaller than the head suggestion (?d ?e ?f) of the list, so
;                      append the head suggestion to the output list ?y and recursively
;                      call itself to process the rest of the suggestion list ?x.
;             Case 3 : The match value ?f of the head suggestion in the column
;                      suggestion list (?d ?e ?f) is smaller than the proposed column
;                      suggestion (?a ?b ?c), so append it (?a ?b ?c) to the output list
;                      ?y, and recursively call itself to process the rest of the
;                      suggestion list ?x using the new lower match (?d ?e ?f).
```

```
            ((findsmallest ?smallest nil ?smallest nil) if (cut))
            ((findsmallest (?a ?b ?c) ((?d ?= ?f).?x) (?1 ?2 ?3) ((?d ?= ?f).?y)) if
                        (<= ?c ?f)
                        (findsmallest (?a ?b ?c) ?x (?1 ?2 ?3) ?y)
            )
            ((findsmallest (?a ?b ?c) ((?d ?= ?f).?x) (?1 ?2 ?3) ((?a ?b ?c).?y)) if
                        (> ?c ?f)
                        (findsmallest (?d ?= ?f) ?x (?1 ?2 ?3) ?y)
            )

; (print_col_list ?col-list) - Print out all of the entries in the ordered
;             column suggestion list ?col-list.
;             See "make_col_list" for the format of the data for column suggestions.
;             Case 1 : Finished printing the column suggestion list.
;             Case 2 : Print out the first suggestion in the column suggestion list.
;                      The "fail" will allow the printing of multiple columns for the
;                      same molecule.
;             Case 3 : Since the head suggestion is printed, call itself to print
;                      the rest of the list.
; NOTE: If multiple column references for the same molecule did not exist,
;             Case 2 & 3 could be combined.
            ((print_col_list nil) if    (cut)
                        (println " + Finished printing column suggestions.....\n")
            )
            ((print_col_list ((?mem ?emb ?value).?col_list)) if
                        (molecule ?mol ?mem ? ?)                    (chiral_col ?emb ?col ?ref)
                        (print " + " ?col " [" [rail ?value] "], matching " ?mol " [")
                          (println ?ref "].")
                        (fail)          ;Will print multiple columns for the same molecule.
            )
            ((print_col_list (?.?col-list)) if
                        (print_col_list ?col-list)
            )

; (removedups (?fa1 ?fa2 ?fa3 ?fa4) (?fb1 ?fb2 ?fb3 ?fb4) ?fla ?flb)   Remove
;             duplicates from the two fragment smiles string lists ?fa1-4 and ?fb1-4.
;             Return two lists that contain the fragments that do not match exactly.
;             Case 1 : Base case, return nil for answer list construction as predicate
;                      recursively returns out.
;             Case 2 : The two head atoms on the fragment lists match, so recursively
;                      call itself with both tails.  Cut used to prevent backtracking.
;                      Since the two match, discard them since they should never be used.
;             Case 3 : The head atom in the first fragment matches a member in the
;                      second fragment list.  Delete it, save head of second list and go
;                      on checking.  Must save head of second list because it hasn't been
;                      checked if it exists in tail of first list.  It will eventually be
;                      eliminated when it's mate from the first list is detected.
;             Case 4 : Reverse of case 3, The head atom in the second fragment matches
;                      a member in the second fragment list.  Delete it, save head of
;                      second list and go on checking.  Must save head of first list
;                      because it hasn't been checked if it exists in tail of second
;                      list.  It will eventually be eliminated when it's mate from the
;                      second list is detected.
;             Case 5 : Both head atoms are unique, so add them to the answers and
;                      recursively call itself to process the remaining tail lists.

            ((removedups nil nil nil nil) if (cut)  )                           ; Case 1
            ((removedups (?h.?t) (?h.?t2) ?fla ?flb) if                         ; Case 2
                        (cut)           (removedups ?t ?t2 ?fla ?flb)
            )
            ((removedups (?h.?t) (?h2.?t2) ?fla ?flb) if                        ; Case 3
                        (deletemember ?h ?t2 ?t3)
                        (cut)           (removedups ?t (?h2.?t3) ?fla ?flb)
            )
            ((removedups (?h.?t) (?h2.?t2) ?fla ?flb) if                        ; Case 4
                        (deletemember ?h2 ?t ?t3)
                        (cut)           (removedups (?h.?t3) ?t2 ?fla ?flb)
            )
            ((removedups (?h.?t) (?h2.?t2) (?h.?fla) (?h2.?flb)) if             ; Case 5
                        (cut)           (removedups ?t ?t2 ?fla ?flb)
            )

; (getinfo chiral ?name) - Build the Chiral Column knowledge base.  It consists of
;             the Smiles string of the molecule, the column name, and a reference.
;             The latter two need to be input by the user.
; NOTE - Should multiple columns be represented by separate entries, or by
;             including all columns, separated by commas??
            ((getinfo_chiral ?name) if
                        (molecule ?name ?mm ? ?)
                        (println "\nSaving " ?name " to the Column knowledge base... ")
                        (print "     What is the Chiral Column used? -> ")
                        (getline stdin ?column)
                        (print "\n     What is the reference for this Column? -> ")
                        (getline stdin ?ref)
                        (not (chiral-col ?mm ?column ?ref))      ;Make sure it doesn't already
                        (open "chirlcol.kb" append chiral_kb)    ;  exist
                        (writeterm chiral_kb ?mm)                       ;Write molecule s-str to kb
                             (write chiral_kb " ")  (writeterm chiral_kb ?column)
                             (write chiral_kb " ")  (writeterm chiral_kb ?ref)
                                        (writeln chiral_kb ".")
                        (println "     Saved " ?name " to the Column knowledge base... \n")
                        (close chiral_kb)
                        (assert ((chiral-col ?mm ?column ?ref)) )
                        (print "     Additional columns to add for ")
                          (print ?name "?\n          (Y/N, default N) -->")
                        (read ?char) (getline stdin ?)
                        (:= ?ans (strcase ?char upper))
                        (not (= "Y" ?ans))
                                    (getinfo_chiral ?name)
                        )           ;END OR
                    )               ;END OR
            )
;   ***END OF assert
)
```

APPENDIX G: Knowledge Bases Used by CHESS

Appendix G lists all of the knowledge bases (KBs) used by CHESS: Molecule KB, Molecular Fragment KB, Fragment Value KB, and Column Suggestion KB. Table XI explains the data contained by each, and the function they serve. Section VII.B covers this information in greater detail.

Molecule Knowledge Base

"MANDELIC ACID" "c(ccc1C(O)C(=O)O)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 O 0) (atom 9 C 0) (atom 10 O 0) (atom 11 O 0))
((bond 2 1 2) (bond 3 2 1) (bond 4 5 1) (bond 5 6 2) (bond 6 1 1) (bond 4 3 2) (bond 7 3 1) (bond 8 7 1) (bond 7 9 1) (bond 9 10 2) (bond 9 11 1))
MANDELAMIDE "c(ccc1C(O)C(=O)N)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 O 0) (atom 9 C 0) (atom 10 O 0) (atom 11 N 0))
((bond 2 1 2) (bond 3 2 1) (bond 4 5 1) (bond 5 6 2) (bond 6 1 1) (bond 4 3 2) (bond 7 3 1) (bond 8 7 1) (bond 7 9 1) (bond 9 10 2) (bond 9 11 1))
"METHYL MANDELATE" "c(ccc1C(O)C(=O)OC)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 O 0) (atom 9 C 0) (atom 10 O 0) (atom 11 O 0) (atom 12 C 0))
((bond 2 1 2) (bond 3 2 1) (bond 4 5 1) (bond 5 6 2) (bond 6 1 1) (bond 4 3 2) (bond 7 3 1) (bond 8 7 1) (bond 7 9 1) (bond 9 10 2) (bond 9 11 1) (bond 11 12 1))
PHENYLGLYCINE "c(ccc1C(N)C(=O)O)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 O 0) (atom 9 O 0) (atom 10 C 3) (atom 11 N 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 7 8 2) (bond 7 9 1) (bond 7 10 1) (bond 10 4 1) (bond 10 11 1))
D-PHENYLALANINE "c(ccc1CC(N)C(=O)O)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 3) (atom 6 C 0) (atom 7 C 0) (atom 8 C 0) (atom 9 N 0) (atom 10 C 0) (atom 11 O 0) (atom 12 O 0))
((bond 1 2 1) (bond 1 3 2) (bond 1 4 1) (bond 2 5 1) (bond 3 6 1) (bond 4 7 2) (bond 5 8 1) (bond 5 9 1) (bond 6 10 2) (bond 8 11 1) (bond 8 12 2) (bond 7 10 1))
"1-PHENYLETHANOL" "c(ccc1C(O)C)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 C 0) (atom 9 O 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 1 7 1) (bond 7 8 1) (bond 7 9 1))
"2-PHENYLPROPIONIC ACID" "c(ccc1C(C(=O)O)C)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 C 0) (atom 9 C 0) (atom 10 O 0) (atom 11 O 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 7 1 1) (bond 8 7 1) (bond 9 7 1) (bond 10 8 2) (bond 8 11 1))
METHYL2-PHENYLPROPIONATE "c(ccc1C(C(=O)OC)C)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 C 0) (atom 9 C 0) (atom 10 O 0) (atom 11 O 0) (atom 12 C 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 7 1 1) (bond 8 7 1) (bond 7 9 1) (bond 10 9 2) (bond 9 11 1) (bond 11 12 1))
"2-AMINO-2-PHENYLETHANOL" "c(ccc1C(N)CO)cc1"
((atom 1 C 3) (atom 2 C 0) (atom 3 O 0) (atom 4 N 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0))
((bond 1 2 1) (bond 2 3 1) (bond 1 4 1) (bond 5 6 1) (bond 6 7 2) (bond 7 8 1) (bond 8 9 2) (bond 9 10 1) (bond 10 5 2) (bond 1 5 1))
"2-AMINO-1-PHENYLETHANOL" "c(ccc1C(O)CN)cc1"
((atom 1 C 0) (atom 2 C 3) (atom 3 O 0) (atom 4 N 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0))
((bond 1 2 1) (bond 2 3 1) (bond 1 4 1) (bond 5 6 1) (bond 6 7 2) (bond 7 8 1) (bond 8 9 2) (bond 9 10 1) (bond 10 5 2) (bond 2 5 1))
"1-PHENYLETHYLAMINE" "c(ccc1C(N)C)cc1"
((atom 1 C 3) (atom 2 C 0) (atom 3 N 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0))
((bond 1 2 1) (bond 1 3 1) (bond 4 5 1) (bond 5 6 2) (bond 6 7 1) (bond 7 8 2) (bond 8 9 1) (bond 9 4 2) (bond 1 4 1))
"1-PHENYL-PROPAN-1-OL" "c(ccc1C(O)CC)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 C 0) (atom 9 O 0) (atom 10 C 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 1 7 1) (bond 7 8 1) (bond 8 10 1))
"ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL" "c(ccc1C(N)C(O)C(-CC=C2)C=C2)cc1"
((atom 1 C 3) (atom 2 C 3) (atom 3 O 0) (atom 4 N 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0) (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 0) (atom 16 C 0))
((bond 1 2 1) (bond 2 3 1) (bond 1 4 1) (bond 5 6 1) (bond 6 7 2) (bond 7 8 1) (bond 8 9 2) (bond 9 10 1) (bond 10 5 2) (bond 1 5 1) (bond 11 12 1) (bond 12 13 2) (bond 13 14 1) (bond 14 15 2) (bond 15 16 1) (bond 16 11 2) (bond 2 11 1))
"2 ACETOXY 3 PHENYLPROPANE" "c(ccc1CC(OC(=O)C)C)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 C 3) (atom 9 C 0) (atom 10 O 0) (atom 11 C 0) (atom 12 O 0) (atom 13 C 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 1 7 1) (bond 7 8 1) (bond 8 10 1) (bond 10 11 1) (bond 11 12 2) (bond 11 13 1))
"2-AMINO-3-METHYLBUTANE" "NC(C)C(C)C"
((atom 1 C 3) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 N 0)) ((bond 1 2 1) (bond 3 1 1) (bond 1 4 1) (bond 2 5 1) (bond 2 6 1))
"2-AMINO-1-HEXANOL" "OCC(N)CCCC"
((atom 1 C 0) (atom 2 C 3) (atom 3 O 0) (atom 4 N 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 C 0))
((bond 1 2 1) (bond 1 3 1) (bond 2 4 1) (bond 5 2 1) (bond 5 6 1) (bond 6 7 1) (bond 7 8 1))
"2-AMINO-1-PENTANOL" "OCC(N)CCC"
((atom 1 C 0) (atom 2 C 3) (atom 3 O 0) (atom 4 N 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0))
((bond 1 2 1) (bond 1 3 1) (bond 2 4 1) (bond 5 2 1) (bond 5 6 1) (bond 6 7 1))
"DL-TYROSINE" "O-c(ccc1CC(N)C(=O)O)cc1"
((atom 1 C 3) (atom 2 C 0) (atom 3 C 0) (atom 4 N 0) (atom 5 O 0) (atom 6 O 0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0) (atom 12 C 0) (atom 13 O 0))
((bond 1 2 1) (bond 1 3 1) (bond 1 4 1) (bond 2 5 1) (bond 2 6 2) (bond 3 7 1) (bond 7 8 2) (bond 7 9 1) (bond 8 10 1) (bond 9 11 2) (bond 10 12 2) (bond 12 13 1) (bond 11 12 1))
"L-3,4-DIHYDROXYPHENYLALANINE" "O-c(ccc1CC(N)C(=O)O)c(O)c1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 O 0) (atom 5 C 0) (atom 6 O 0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 3) (atom 11 C 0) (atom 12 N 0) (atom 13 O 0) (atom 14 O 0))
((bond 1 2 2) (bond 1 3 1) (bond 1 4 1) (bond 2 5 1) (bond 2 6 1) (bond 3 7 2) (bond 5 8 2) (bond 7 9 1) (bond 9 10 1) (bond 10 11 1) (bond 10 12 1) (bond 11 13 1) (bond 11 14 2) (bond 7 8 1))
"D-(+)-TRYPTOPHAN" "c(ccc1C=2CC(N)C(=O)O)cc1NC2"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 N 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 3) (atom 11 C 0) (atom 12 C 0) (atom 13 N 0) (atom 14 O 0) (atom 15 O 0))
((bond 1 2 1) (bond 1 3 2) (bond 1 4 1) (bond 2 5 1) (bond 2 6 2) (bond 3 7 1) (bond 3 8 1) (bond 4 9 2) (bond 5 10 1) (bond 8 11 2) (bond 10 12 1) (bond 10 13 1) (bond 12 14 1) (bond 12 15 2) (bond 6 7 1) (bond 9 11 1))
"L-GLUTAMINE" "O=C(N)CCC(N)C(=O)O"
((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O 0) (atom 7 C 0) (atom 8 C 0) (atom 9 N 0) (atom 10 O 0))
((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7 1) (bond 7 8 1) (bond 8 9 1) (bond 8 10 2))
"2 BUTYL PHENYL CARBAMATE" "c(ccc1NC(=O)OC(C)CC)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 N 0) (atom 8 C 0) (atom 9 O 0) (atom 10 O 0) (atom 11 C 3) (atom 12 C 0) (atom 13 C 0) (atom 14 C 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 2 7 1) (bond 8 9 2) (bond 8 10 1) (bond 10 11 1) (bond 7 8 1) (bond 11 12 1) (bond 13 12 1) (bond 11 14 1))
"2-PHENYL-PROPAN-1-OL" "c(ccc1C(C)CO)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 C 0) (atom 9 O 0) (atom 10 C 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 7 8 1) (bond 8 9 1) (bond 7 10 1))
"1-PHENYL-1,2-ETHANEDIOL" "c(ccc1C(O)CO)cc1"
((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 3) (atom 8 C 0) (atom 9 O 0) (atom 10 O 0))
((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1

```
        2) (bond 1 7 1) (bond 7 8 1) (bond 8 9 1) (bond 7 10 1))
"1-PHENYL-1,3-PROPANEDIOL" "c(ccc1C(O)CO)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 3) (atom 8 C 0) (atom 9 O 0) (atom 10 C 0) (atom 11 O 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 1 7 1) (bond 7 8 1) (bond 7 9 1) (bond 8 10 1) (bond 10 11 1))
"N,O-DIBENZOYL2-AMINOBUTANOL" "c(ccc1C(=O)OCC(NC(=O)C(-CC=C2)C=C2)CC)cc1"
   ((atom 1 C 3) (atom 2 N 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 O 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 0) (atom 16 O 0)
      (atom 17 C 0) (atom 18 C 0) (atom 19 C 0) (atom 20 C 0) (atom 21 C 0)
      (atom 22 C 0))
   ((bond 1 2 1) (bond 3 1 1) (bond 3 4 1) (bond 5 1 1) (bond 6 5 1) (bond 7 6
        1) (bond 8 7 2) (bond 9 10 1) (bond 10 11 2) (bond 11 12 1) (bond 12
        13 2) (bond 13 14 1) (bond 14 9 2) (bond 7 9 1) (bond 15 2 1)
      (bond 16 15 2) (bond 17 18 1) (bond 18 19 2) (bond 19 20 1) (bond 20 21 2)
      (bond 21 22 1) (bond 22 17 2) (bond 15 22 1))
"ALPHA METHYL PHENETHYL BENZOATE" "c(ccc1CC(OC(=O)C(=CC-C2)C=C2)C)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 C 3) (atom 9 C 0) (atom 10 O 0) (atom 11 C 0)
      (atom 12 O 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 0) (atom 16 C 0)
      (atom 17 C 0) (atom 18 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 1 7 1) (bond 7 8 1) (bond 9 8 1) (bond 8 10 1) (bond 10 11 1)
      (bond 11 12 2) (bond 13 14 1) (bond 14 15 2) (bond 15 16 1) (bond 16 17 2)
      (bond 17 18 1) (bond 18 13 2) (bond 11 13 1))
"1,3-DIACETOXY-BUTANE" "O=C(OCCC(OC(=O)C)C)C"
   ((atom 1 C 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 C
        0) (atom 7 O 0) (atom 8 C 0) (atom 9 O 0) (atom 10 C 0) (atom 11 O 0)
      (atom 12 C 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 3 4 1) (bond 4 5 1) (bond 5 6 1) (bond 7 6
        2) (bond 6 8 1) (bond 9 2 1) (bond 9 10 1) (bond 11 10 2) (bond 10 12
        1))
PRACTOLOL "c(cc(c1)OCC(O)CNC(C)C)c(c1)NC(=O)C"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 N 0) (atom 8 C 0) (atom 9 O 0) (atom 10 C 0) (atom 11 O 0)
      (atom 12 C 0) (atom 13 C 3) (atom 14 C 0) (atom 15 O 0) (atom 16 N 0)
      (atom 17 C 3) (atom 18 C 0) (atom 19 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 7 4 1) (bond 7 8 1) (bond 8 9 2) (bond 8 10 1) (bond 1 11 1)
      (bond 11 12 1) (bond 12 13 1) (bond 13 14 1) (bond 13 15 1) (bond 14 16 1)
      (bond 16 17 1) (bond 17 18 1) (bond 17 19 1))
BENZOIN "c(ccc1C(=O)C(O)C(-CC=C2)C=C2)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 O 0) (atom 9 C 3) (atom 10 O 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 0) (atom 16 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 1 7 1) (bond 7 8 2) (bond 7 9 1) (bond 10 9 1) (bond 11 12 1)
      (bond 12 13 2) (bond 13 14 1) (bond 14 15 2) (bond 15 16 1) (bond 16 11 2)
      (bond 9 11 1))
CARTEOLOL "O=C(N-c(c1cc2)c(c2)OCC(O)CN(C)C)CC1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 N 0) (atom 11 O 0)
      (atom 12 O 0) (atom 13 C 0) (atom 14 C 3) (atom 15 C 0) (atom 16 O 0)
      (atom 17 N 0) (atom 18 C 0) (atom 19 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 7 8 1) (bond 8 9 1) (bond 9 4 1) (bond 10 3 1) (bond 7 10 1)
      (bond 11 7 2) (bond 12 2 1) (bond 12 13 1) (bond 13 14 1) (bond 14 15 1)
      (bond 14 16 1) (bond 15 17 1) (bond 17 18 1) (bond 17 19 1))
CYCLOPENTOLATE "c(ccc1C(C(=O)OCCN(C)C)C(CC2)(CC2)O)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 3) (atom 8 C 3) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 O 0) (atom 14 C 0) (atom 15 O 0) (atom 16 O 0)
      (atom 17 C 0) (atom 18 C 0) (atom 19 N 0) (atom 20 C 0) (atom 21 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 2 7 1) (bond 7 8 1) (bond 8 9 1) (bond 9 10 1) (bond 10 11 1)
      (bond 11 12 1) (bond 12 8 1) (bond 8 13 1) (bond 7 14 1) (bond 14 15 1)
      (bond 16 14 2) (bond 15 17 1) (bond 17 18 1) (bond 18 19 1) (bond 19 20 1)
      (bond 19 21 1))
HISTIDINE "N(=CN1)C=C1CC(N)C(=O)O"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 N 0) (atom 4 C 0) (atom 5 N 0) (atom 6 C
        0) (atom 7 C 3) (atom 8 N 0) (atom 9 C 0) (atom 10 O 0) (atom 11 O 0))
   ((bond 1 2 2) (bond 1 3 1) (bond 3 4 2) (bond 5 4 1) (bond 5 2 1) (bond 2 6
        1) (bond 6 7 1) (bond 8 7 1) (bond 9 10 2) (bond 9 11 1) (bond 7 9 1))
METHYL-3-BENZOYLOXYPROPANOATE "c(ccc1C(=O)OC(C)CC(=O)OC)cc1"
   ((atom 1 C 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 O 0) (atom 9 C 0) (atom 10 O 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 0) (atom 16 C 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 3 4 1) (bond 5 4 2) (bond 4 6 1) (bond 6 7
        1) (bond 8 2 1) (bond 8 9 1) (bond 10 9 2) (bond 11 12 1) (bond 12 13
        2) (bond 13 14 1) (bond 14 15 2) (bond 15 16 1) (bond 16 11 2)
      (bond 9 11 1))
GUALFENESINE "c(ccc1OCC(O)CO)cc1OC"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 O 0) (atom 8 O 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 3)
      (atom 12 C 0) (atom 13 O 0) (atom 14 O 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 1 7 1) (bond 6 8 1) (bond 8 9 1) (bond 7 10 1) (bond 10 11 1)
      (bond 11 12 1) (bond 13 11 1) (bond 12 14 1))
"1-(1-HYDROXYETHYL)NAPHTHALENE" "c(ccc1C(=C2)C(O)C)cc1C=C2"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 3)
      (atom 12 C 0) (atom 13 O 0))
   ((bond 1 2 2) (bond 1 3 1) (bond 1 4 1) (bond 2 5 1) (bond 2 6 1) (bond 3 7
        2) (bond 4 8 2) (bond 5 9 2) (bond 6 10 2) (bond 7 9 1) (bond 8 10 1)
      (bond 6 11 1) (bond 11 12 1) (bond 11 13 1))
"2-(1-HYDROXYETHYL)NAPHTHALENE" "c(ccc1C=C2C(O)C)cc1C=C2"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 3)
      (atom 12 C 0) (atom 13 O 0))
   ((bond 1 2 2) (bond 1 3 1) (bond 1 4 1) (bond 2 5 1) (bond 2 6 1) (bond 3 7
        2) (bond 4 8 2) (bond 5 9 2) (bond 6 10 2) (bond 7 9 1) (bond 8 10 1)
      (bond 11 12 1) (bond 11 13 1) (bond 11 10 1))
"9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE"
"c(ccc1C=C2C-C3)cc1C(-C2C=C3)C(O)C(H)(H)H"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 3) (atom 16 O 0)
      (atom 17 C 3) (atom 18 F 0) (atom 19 F 0) (atom 20 F 0))
   ((bond 1 2 2) (bond 1 3 1) (bond 1 4 1) (bond 2 5 1) (bond 2 6 1) (bond 3 7
        2) (bond 4 8 2) (bond 5 9 2) (bond 6 10 2) (bond 7 9 1) (bond 8 10 1)
      (bond 11 12 2) (bond 12 13 1) (bond 13 14 2) (bond 11 8 1) (bond 10 14 1)
      (bond 15 6 1) (bond 15 16 1) (bond 15 17 1) (bond 19 17 1) (bond 18 17 1)
      (bond 20 17 1))
CARBINOXAMINE "Cl-c(ccc1C(OCCN(C)C)C(N=CC=2)=CC2)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 Cl 0) (atom 8 C 3) (atom 9 O 0) (atom 10 C 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 N 0) (atom 16 C 0)
      (atom 17 C 0) (atom 18 N 0) (atom 19 C 0) (atom 20 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 4 7 1) (bond 1 8 1) (bond 8 9 1) (bond 10 11 2) (bond 11 12 1)
      (bond 12 13 2) (bond 13 14 1) (bond 8 10 1) (bond 15 14 2) (bond 15 10 1)
      (bond 9 16 1) (bond 16 17 1) (bond 17 18 1) (bond 18 19 1) (bond 18 20 1))
"4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID" "Cl-c(ccc1C(CN)CC(=O)O)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 Cl 0) (atom 8 C 3) (atom 9 C 0) (atom 10 N 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 O 0) (atom 14 O 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 5 7 1) (bond 2 8 1) (bond 8 9 1) (bond 9 10 1) (bond 8 11 1)
      (bond 12 13 2) (bond 12 14 1) (bond 11 12 1))
SULCONAZOL "Cl-c(cc(Cl)c1C(SCC(=CC=C2)C=C2)CN(C=N3)C=C3)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 Cl 0) (atom 8 Cl 0) (atom 9 C 0) (atom 10 N 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 N 0) (atom 14 C 0) (atom 15 C 0) (atom 16 C 0)
      (atom 17 C 0) (atom 18 C 0) (atom 19 C 0) (atom 20 C 0) (atom 21 C 0)
      (atom 22 C 0) (atom 23 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 7 3 1) (bond 1 8 1) (bond 6 9 1) (bond 9 11 1) (bond 10 12 1)
```

```
      (bond 12 13 2) (bond 10 14 1) (bond 14 15 2) (bond 15 13 1) (bond 11 16 1)
      (bond 17 18 1) (bond 18 19 2) (bond 19 20 1) (bond 20 21 2) (bond 21 22 1)
      (bond 22 17 2) (bond 16 20 1) (bond 9 23 1) (bond 23 10 1))
"FLURBIPROFEN ME ESTER" "C(C(=O)OC)c1cc(-c2ccccc2)c(F)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0)
      (atom 12 F 0) (atom 13 C 3) (atom 14 C 0) (atom 15 C 0) (atom 16 O 0)
      (atom 17 O 0) (atom 18 C 0) (atom 19 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 7 8 2) (bond 8 9 1) (bond 9 10 2) (bond 10 11 1) (bond 5 7 1)
      (bond 12 4 1) (bond 2 13 1) (bond 13 14 1) (bond 13 15 1) (bond 15 17 1)
      (bond 15 16 2) (bond 17 18 1) (bond 7 19 1) (bond 11 19 2))
"BENZYL PHENYL CARBINOL" "c(ccc1C(O)CC(-CC-C2)C-C2)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 0))
   ((bond 1 2 1) (bond 1 3 2) (bond 1 4 1) (bond 2 5 1) (bond 2 6 1) (bond 3 7
        1) (bond 4 8 2) (bond 5 9 1) (bond 7 10 2) (bond 9 11 2) (bond 9 12 1)
      (bond 11 13 1) (bond 12 14 2) (bond 13 15 2) (bond 8 10 1) (bond 14 15 1))
PROPRANOLOL "C(=CC=C1)-c(ccc2)c1c20CC(O)CNC(C)C"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 O 0)
      (atom 12 C 0) (atom 13 C 3) (atom 14 C 0) (atom 15 O 0) (atom 16 N 0)
      (atom 17 C 3) (atom 18 C 0) (atom 19 C 0))
   ((bond 1 2 2) (bond 2 3 1) (bond 3 4 2) (bond 4 5 1) (bond 5 6 2) (bond 6 1
        1) (bond 2 7 1) (bond 7 8 2) (bond 8 9 1) (bond 9 10 2) (bond 10 1 1)
      (bond 11 3 1) (bond 11 12 1) (bond 12 13 1) (bond 13 14 1) (bond 13 15 1)
      (bond 16 14 1) (bond 16 17 1) (bond 17 18 1) (bond 17 19 1))
PINDOLOL "N(C=C1)-c(ccc2)c1c20CC(O)CNC(C)C"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 N 0) (atom 9 O 0) (atom 10 C 0) (atom 11 C 3)
      (atom 12 C 0) (atom 13 O 0) (atom 14 N 0) (atom 15 C 3) (atom 16 C 0)
      (atom 17 C 0) (atom 18 C 0))
   ((bond 1 2 2) (bond 2 3 1) (bond 3 4 2) (bond 4 5 1) (bond 5 6 2) (bond 6 1
        1) (bond 8 1 1) (bond 7 3 1) (bond 9 10 1) (bond 10 11 1) (bond 11 12
        1) (bond 11 13 1) (bond 14 12 1) (bond 14 15 1) (bond 15 16 1)
      (bond 15 17 1) (bond 2 7 1) (bond 8 18 1) (bond 18 7 2))
METOPROLOL "c(cc(c1)OCC(O)CNC(C)C)c(c1)CCOC"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 O 0) (atom 8 C 0) (atom 9 C 3) (atom 10 C 0) (atom 11 O 0)
      (atom 12 N 0) (atom 13 C 3) (atom 14 C 0) (atom 15 C 0) (atom 16 C 0)
      (atom 17 C 0) (atom 18 O 0) (atom 19 C 0))
   ((bond 1 2 2) (bond 2 3 1) (bond 3 4 2) (bond 4 5 1) (bond 5 6 2) (bond 6 1
        1) (bond 7 3 1) (bond 7 8 1) (bond 8 9 1) (bond 9 10 1) (bond 9 11 1)
      (bond 12 10 1) (bond 12 13 1) (bond 13 14 1) (bond 13 15 1) (bond 6 16 1)
      (bond 16 17 1) (bond 17 18 1) (bond 18 19 1))
ALPRENOLOL "c(ccc1OCC(O)CNC(C)C)cc1CC=C"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 3) (atom 9 C 0) (atom 10 O 0) (atom 11 N 0)
      (atom 12 C 3) (atom 13 C 0) (atom 14 C 0) (atom 15 C 0) (atom 16 C 0)
      (atom 17 C 0) (atom 18 C 0))
   ((bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 1 1) (bond 6 2 1) (bond 6 7
        1) (bond 7 8 1) (bond 8 9 1) (bond 8 10 1) (bond 11 9 1) (bond 11 12 1)
      (bond 12 13 1) (bond 12 14 1) (bond 2 15 1) (bond 1 15 2) (bond 15 16 1)
      (bond 16 17 1) (bond 17 18 2))
PERISOXAL "c(ccc1C(ON-2)-CC2C(O)CN(CCC3)CC3)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 N 0) (atom 8 O 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0)
      (atom 12 C 3) (atom 13 C 0) (atom 14 O 0) (atom 15 C 0) (atom 16 C 0)
      (atom 17 C 0) (atom 18 N 0) (atom 19 C 0) (atom 20 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 7 8 1) (bond 7 9 2) (bond 8 10 1) (bond 9 11 1) (bond 10 11 2)
      (bond 1 10 1) (bond 9 12 1) (bond 12 13 1) (bond 14 12 1) (bond 15 16 1)
      (bond 16 17 1) (bond 17 18 1) (bond 18 19 1) (bond 19 20 1) (bond 20 15 1)
      (bond 13 18 1))
L-METHIONINE "S(C)CCC(N)C(=O)O"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 7 8 1) (bond 8 9 1))
LYSINE "O=C(O)C(N)CCCCN"
   ((atom 1 N 0) (atom 2 C 4) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 N 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 7 8 1) (bond 8 9 1) (bond 9 10 1))
CITRULLINE "O=C(N)NCCCC(N)C(=O)O"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 N 0) (atom 10 C 0) (atom 11 O 0)
      (atom 12 N 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 7 8 1) (bond 8 9 1) (bond 9 10 1) (bond 10 11 2) (bond 10 12
        1))
DL VALINE "O=C(O)C(N)C(C)C"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 3) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 3 8 1))
L-ISOLEUCINE "O=C(O)C(N)C(C)CC"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 3) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 7 8 1) (bond 3 9 1))
L-ORNITHINE "O=C(O)C(N)CCCN"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 N 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 7 8 1) (bond 8 9 1))
ARGININE "O=C(O)C(N)CCCNC(=N)N"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 N 0) (atom 10 C 0) (atom 11 N 0)
      (atom 12 N 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 7 8 1) (bond 8 9 1) (bond 9 10 1) (bond 10 11 1) (bond 10 12
        2))
"L-GLUTAMIC ACID" "O-C(O)CCC(N)C(=O)O"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 0) (atom 8 C 0) (atom 9 O 0) (atom 10 O 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 7 8 1) (bond 8 9 1) (bond 8 10 2))
ASPARAGINE "O=C(N)CC(N)C(=O)O"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 O 0) (atom 5 O 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 N 0) (atom 9 O 0))
   ((bond 1 2 1) (bond 3 4 2) (bond 3 5 1) (bond 2 3 1) (bond 6 7 1) (bond 7 8
        1) (bond 7 9 2) (bond 2 6 1))
"ASPARTIC ACID" "O=C(O)CC(N)C(=O)O"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 O 0) (atom 5 O 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 O 0) (atom 9 O 0))
   ((bond 1 2 1) (bond 3 4 2) (bond 3 5 1) (bond 2 3 1) (bond 6 7 1) (bond 7 8
        1) (bond 7 9 2) (bond 2 6 1))
DL-LEUCINE "O=C(O)C(N)CC(C)C"
   ((atom 1 N 0) (atom 2 C 3) (atom 3 C 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O
        0) (atom 7 C 3) (atom 8 C 0) (atom 9 C 0))
   ((bond 1 2 1) (bond 2 3 1) (bond 4 5 2) (bond 4 6 1) (bond 2 4 1) (bond 3 7
        1) (bond 7 8 1) (bond 7 9 1))
ACETYLPHENETHOLIDE "c(ccc1C(C(=O)NC(=O)NC(=O)C)CC)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 3) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 O 0)
      (atom 12 N 0) (atom 13 C 0) (atom 14 O 0) (atom 15 N 0) (atom 16 C 0)
      (atom 17 O 0) (atom 18 C 0))
   ((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1
        2) (bond 1 7 1) (bond 7 8 1) (bond 8 9 1) (bond 10 11 2) (bond 10 12 1)
      (bond 7 10 1) (bond 13 14 2) (bond 14 15 1) (bond 12 13 1) (bond 15 17 2)
      (bond 16 18 1) (bond 15 16 1))
KETOPROFEN "c(ccc1C(=O)C(C=CC=2)=CC2C(C(=O)O)C)cc1"
   ((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C
        0) (atom 7 C 0) (atom 8 O 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0)
      (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 3) (atom 16 C 0)
      (atom 17 C 0) (atom 18 O 0) (atom 19 O 0))
```

((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 7 1 1) (bond 8 7 2) (bond 9 10 1) (bond 10 11 2) (bond 11 12 1) (bond 12 13 2) (bond 13 14 1) (bond 14 9 2) (bond 7 9 1) (bond 15 13 1) (bond 16 15 1) (bond 17 15 1) (bond 17 18 2) (bond 17 19 1))

FENOPROFEN "c(ccc1OC(C=CC-2)=CC2C(C(=O)O)C)cc1"

((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0) (atom 12 C 0) (atom 13 C 3) (atom 14 C 0) (atom 15 C 0) (atom 16 O 0) (atom 17 O 0) (atom 18 O 0))

((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 7 8 1) (bond 8 9 2) (bond 9 10 1) (bond 10 11 2) (bond 11 12 1) (bond 12 7 2) (bond 13 11 1) (bond 14 13 1) (bond 15 13 1) (bond 15 16 2) (bond 15 17 1) (bond 18 1 1) (bond 18 7 1))

"LACTIC ACID" "O=C(O)C(O)C" ((atom 1 C 3) (atom 2 C 0) (atom 3 O 0) (atom 4 C 0) (atom 5 O 0) (atom 6 O 0)) ((bond 1 2 1) (bond 3 1 1) (bond 4 5 2) (bond 4 6 1) (bond 1 4 1))

"GLYCERIC ACID" "O=C(O)C(O)CO"

((atom 1 C 0) (atom 2 C 3) (atom 3 O 0) (atom 4 O 0) (atom 5 C 0) (atom 6 O 0) (atom 7 O 0))

((bond 1 2 1) (bond 3 1 1) (bond 4 2 1) (bond 5 6 2) (bond 5 7 1) (bond 2 5 1))

"ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE" "O-c(ccc1OC((C(=O)OCC)C)cc1"

((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 O 0) (atom 8 O 0) (atom 9 C 3) (atom 10 C 0) (atom 11 C 0) (atom 12 O 0) (atom 13 O 0) (atom 14 C 0) (atom 15 C 0))

((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 7 4 1) (bond 1 8 1) (bond 8 9 1) (bond 9 10 1) (bond 9 11 1) (bond 11 12 1) (bond 13 11 2) (bond 12 14 1) (bond 14 15 1))

WARFARINE "c(ccc1C(O)=C2C(C(=CC=C3)C=C3)CC(=O)C)cc1OC2=O"

((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 O 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 O 0) (atom 12 O 0) (atom 13 C 3) (atom 14 C 0) (atom 15 C 0) (atom 16 C 0) (atom 17 C 0) (atom 18 C 0) (atom 19 C 0) (atom 20 C 0) (atom 21 C 0) (atom 22 C 0) (atom 23 O 0))

((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 1 7 1) (bond 7 8 1) (bond 8 9 1) (bond 9 10 2) (bond 10 6 1) (bond 8 11 2) (bond 10 12 1) (bond 9 13 1) (bond 13 14 1) (bond 14 15 1) (bond 15 16 1) (bond 17 18 1) (bond 18 19 2) (bond 19 20 1) (bond 20 21 2) (bond 21 22 1) (bond 22 17 2) (bond 13 18 1) (bond 15 23 2))

N-AMIDE-L-PHENYLALANINE "c(ccc1CC(NC(=O)N)C(=O)O)cc1"

((atom 1 C 3) (atom 2 C 0) (atom 3 N 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 O 0) (atom 8 O 0) (atom 9 C 0) (atom 10 C 0) (atom 11 O 0) (atom 12 C 0) (atom 13 C 0) (atom 14 C 0) (atom 15 N 0))

((bond 1 2 1) (bond 1 3 1) (bond 1 4 1) (bond 2 5 1) (bond 3 6 1) (bond 4 7 1) (bond 4 8 2) (bond 5 9 2) (bond 5 10 1) (bond 6 11 2) (bond 9 12 1) (bond 10 13 2) (bond 12 14 2) (bond 13 14 1) (bond 6 15 1))

"N-ACETYL TRYPTOPHAN" "c(ccc1C-2CC(NC(=O)C)C(=O)O)cc1NC2"

((atom 1 C 3) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 N 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0) (atom 12 C 0) (atom 13 O 0) (atom 14 O 0) (atom 15 N 0) (atom 16 C 0) (atom 17 O 0) (atom 18 C 0))

((bond 1 2 1) (bond 2 3 1) (bond 3 4 2) (bond 3 5 1) (bond 5 6 2) (bond 4 7 1) (bond 6 7 1) (bond 5 8 1) (bond 8 9 2) (bond 9 10 1) (bond 10 11 2) (bond 11 6 1) (bond 12 13 2) (bond 12 14 1) (bond 1 12 1) (bond 1 15 1) (bond 15 16 1) (bond 16 17 2) (bond 16 18 1))

"N-ACTEYL VALINE" "O=C(NC(C(=O)O)C(C)C)C"

((atom 1 C 3) (atom 2 C 0) (atom 3 O 0) (atom 4 O 0) (atom 5 N 0) (atom 6 C 0) (atom 7 O 0) (atom 8 C 0) (atom 9 C 3) (atom 10 C 0) (atom 11 C 0))

((bond 2 3 2) (bond 2 4 1) (bond 1 2 1) (bond 1 5 1) (bond 5 6 1) (bond 6 7 2) (bond 6 8 1) (bond 9 10 1) (bond 9 11 1) (bond 1 9 1))

TRIADIMEFON "Cl c(ccc1OC(N(C N2)C=C2)C(=O)C(C)(C)C)cc1"

((atom 1 C 0) (atom 2 N 0) (atom 3 C 0) (atom 4 C 0) (atom 5 N 0) (atom 6 C 3) (atom 7 O 0) (atom 8 C 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0) (atom 12 C 0) (atom 13 C 0) (atom 14 Cl 0) (atom 15 C 0) (atom 16 C 3) (atom 17 O 0) (atom 18 C 0) (atom 19 C 0) (atom 20 C 0))

((bond 1 3 2) (bond 2 4 1) (bond 3 5 1) (bond 4 5 2) (bond 1 2 1) (bond 2 6 1) (bond 7 6 1) (bond 8 7 1) (bond 9 10 2) (bond 10 11 1) (bond 11 12 2) (bond 12 13 1) (bond 13 8 2) (bond 7 8 1) (bond 14 11 1) (bond 6 15 1) (bond 15 16 1) (bond 17 15 2) (bond 16 18 1) (bond 16 19 1) (bond 16 20 1))

"TIAPROFENIC ACID ME ESTER" "S(C(=C1)C(C(=O)OC)C)C(=C1)C(=O)-c(ccc2)cc2"

((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 C 0) (atom 8 O 0) (atom 9 C 0) (atom 10 S 0) (atom 11 C 0) (atom 12 C 0) (atom 13 C 0) (atom 14 C 3) (atom 15 C 0) (atom 16 C 0) (atom 17 O 0) (atom 18 O 0) (atom 19 C 0))

((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 5 7 1) (bond 7 8 2) (bond 9 10 1) (bond 9 11 2) (bond 10 12 1) (bond 11 13 1) (bond 12 13 2) (bond 7 9 1) (bond 12 14 1) (bond 14 15 1) (bond 14 16 1) (bond 15 17 1) (bond 15 18 2) (bond 17 19 1))

"1-BROMO ETHYLBENZENE" "BrC(-c(ccc1)cc1)C"

((atom 1 C 0) (atom 2 C 0) (atom 3 C 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 Br 0) (atom 8 C 3) (atom 9 C 0))

((bond 1 2 1) (bond 2 3 2) (bond 3 4 1) (bond 4 5 2) (bond 5 6 1) (bond 6 1 2) (bond 1 8 1) (bond 8 7 1) (bond 8 9 1))

N-ACETYL-L-PHENYLALANINE "c(ccc1CC(NC(=O)C)C(=O)O)cc1"

((atom 1 C 3) (atom 2 C 0) (atom 3 N 0) (atom 4 C 0) (atom 5 C 0) (atom 6 C 0) (atom 7 O 0) (atom 8 O 0) (atom 9 C 0) (atom 10 C 0) (atom 11 C 0) (atom 12 O 0) (atom 13 C 0) (atom 14 C 0) (atom 15 C 0))

((bond 1 2 1) (bond 1 3 1) (bond 1 4 1) (bond 2 5 1) (bond 3 6 1) (bond 4 7 1) (bond 4 8 2) (bond 5 9 2) (bond 5 10 1) (bond 6 11 1) (bond 6 12 2) (bond 9 13 1) (bond 10 14 2) (bond 13 15 2) (bond 14 15 1))

Molecular Fragment Knowledge Base

"c(ccc1C(O)C(=O)O)cc1" "c(ccc1)cc1" O=CO O ""

"c(ccc1C(O)C(=O)N)cc1" "c(ccc1)cc1" O=CN O ""

"c(ccc1C(O)C(=O)OC)cc1" "c(ccc1)cc1" O=COC O ""

"c(ccc1C(N)C(=O)O)cc1" "c(ccc1)cc1" O=CO N ""

"c(ccc1CC(N)C(=O)O)cc1" "c(ccc1C)cc1" O=CO N ""

"c(ccc1C(O)C)cc1" "c(ccc1)cc1" C O ""

"c(ccc1C(C(-O)O)C)cc1" "c(ccc1)cc1" O=CO C ""

"c(ccc1C(C(-O)OC)C)cc1" "c(ccc1)cc1" O=COC C ""

"c(ccc1C(N)CO)cc1" "c(ccc1)cc1" OC N ""

"c(ccc1C(O)CN)cc1" "c(ccc1)cc1" NC O ""

"c(ccc1C(N)C)cc1" "c(ccc1)cc1" C N ""

"c(ccc1C(O)CC)cc1" "c(ccc1)cc1" CC O ""

"c(ccc1C(N)C(O)C(=CC=C2)C=C2)cc1" "c(ccc1CO)cc1" "c(ccc1)cc1" N ""

"c(ccc1CC(OC(=O)C)C)cc1" "c(ccc1C)cc1" "O=C(O)C" C ""

"NC(C)C(C)C" NCC C C ""

"OCC(N)CCCC" CCCC OC N ""

"OCC(N)CCC" CCC OC N ""

"O-c(ccc1CC(N)C(=O)O)cc1" "c(cc(c1)C)c(c1)O" O=CO N ""

"O-c(ccc1CC(N)C(=O)O)c(O)c1" "c(cc(O)c1O)c(c1)C" O=CO N ""

"c(ccc1C-2CC(N)C(=O)O)cc1NC2" "c(ccc1C=2C)cc1NC2" O=CO N ""

"O=C(N)CCC(N)C(=O)O" "O=C(N)CC" O=CO N ""

"c(ccc1NC(=O)OC(C)CC)cc1" "c(ccc1NC(=O)O)cc1" CC C ""

"c(ccc1C(C)CO)cc1" "c(ccc1)cc1" OC C ""

"c(ccc1C(O)CO)cc1" "c(ccc1)cc1" OC O ""

"c(ccc1C(O)CCO)cc1" "c(ccc1)cc1" OCC O ""

"c(ccc1C(=O)OCC(NC(=O)C(=CC=C2)C=C2)CC)cc1" "c(ccc1C(=O)OC)cc1" "c(ccc1C(=O)N)cc1" CC ""

"c(ccc1CC(OC(=O)C(=CC=C2)C=C2)C)cc1" "c(ccc1C(=O)O)cc1" "c(ccc1C)cc1" C ""

"O=C(OCCC(OC(=O)C)C)C" "O=C(OCC)C" "O=C(O)C" C ""

"c(cc(c1)OCC(O)CNC(C)C)c(c1)NC(=O)C" "c(cc(c1)NC(=O)C)c(c1)OC" "N(C)C(C)C" O ""

"c(ccc1C(=O)C(O)C(=CC=C2)C=C2)cc1" "c(ccc1C=O)cc1" "c(ccc1)cc1" O ""

"O=C(N=c(c1cc2)c(n2)OCC(O)CN(C)C)CC1" "c(cc(OC)c1NC2=O)cc1CC2" "CN(C)C" O ""

"c(ccc1C(C(=O)OCCN(C)C)C(CC2)(CC2)O)cc1" "O=C(OCCN(C)C)" "c(ccc1)cc1" "OC(CC1)(CC1" ""

"N(=CN1)C=C1CC(N)C(=O)O" "N(=CN1)C=C1C" O=CO N ""

"c(ccc1C(=O)OC(C)CC(=O)OC)cc1" "c(ccc1C(=O)O)cc1" "O=C(OC)C" C ""

"c(ccc1OCC(O)CO)cc1OC" "c(ccc1OC)cc1OC" OC O ""

"c(ccc1C(=C2)C(O)C)cc1C=C2" "c(ccc1C=C2)cc1C=C2" C O ""

"c(ccc1C=C2C(O)C)cc1C=C2" "c(ccc1C=C2)cc1C=C2" C O ""

"c(ccc1C=C2C=C3)cc1C(=C2C=C3)C(O)C(H)(H)H" "c(ccc1C=C2C=C3)cc1C=C2C=C3" "C(H)(H)H" O ""

"Cl-c(ccc1C(OCCN(C)C)C(N(CC=2)-CC2)cc1" "Cl c(ccc1)cc1" "OCCN(C)C" "N(C=CC-1)-CC1"
""
"Cl-c(ccc1C(CN)CC(=O)O)cc1" "Cl c(ccc1)cc1" "O=C(O)C" NC ""
"Cl c(cc(Cl)c1C(SCC(=CC=C2)C=C2)CN(C=N3)C=C3)cc1" "Cl-c(ccc1)cc1Cl"
"SC c(ccc1)cc1"
"N(=CN1C)C=C1" ""
"C(C=CC=1)=C(C1)-c(ccc2OC(C(=O)OC)C)c(F)c2" "c(ccc1C(C=CC 2)=C(F)C2)cc1" O=COC C
""
"c(ccc1C(O)CC(=CC=C2)C=C2)cc1" "c(ccc1C)cc1" "c(ccc1)cc1" O ""
"C(=CC=C1)-c(ccc2)c1c2OCC(O)CNC(C)C" "c(ccc1C(=C2)OC)cc1C=C2" "N(C)C(C)C" O ""
"N(C=C1)-c(ccc2)c1c2OCC(O)CNC(C)C" "N(C=C1)-c(ccc2)c1c2OC" "N(C)C(C)C" O ""
"c(cc(c1)OCC(O)CNC(C)C)c(c1)COC" "c(cc(c1)COC)c(c1)OC" "N(C)C(C)C" O ""
"c(ccc1OCC(O)CNC(C)C)cc1CC=C" "c(ccc1CC=C)cc1OC" "N(C)C(C)C" O ""
"c(ccc1C(ON=2)=CC2C(O)CN(CCC3)CC3)cc1" "O(N=C1)C(=C1)-c(ccc2)cc2" "C(CCC1)N(C1)C"
O ""
"S(C)CCC(N)C(=O)O" "S(C)CC" O=CO N ""
"O=C(O)C(N)CCCCN" NCCCC O=CO N ""
"O=C(N)NCCCC(N)C(=O)O" "O=C(N)NCCC" O=CO N ""
"O=C(O)C(N)C(C)C" CCC O=CO N ""
"O=C(O)C(N)C(C)CC" CCCC O=CO N ""
"O=C(O)C(N)CCCN" NCCC O=CO N ""
"O=C(O)C(N)CCCNC(=N)N" "N=C(N)NCCC" O=CO N ""
"O=C(O)CCC(N)C(=O)O" "O=C(O)CC" O=CO N ""
"O=C(N)CC(N)C(=O)O" "O=C(N)C" O=CO N ""
"O=C(O)CC(N)C(=O)O" "O=C(O)C" O=CO N ""
"O=C(O)C(N)CC(C)C" "CC(C)C" O=CO N ""
"c(ccc1C(C(=O)NC(=O)NC(=O)C)CC)cc1" "O=CNC(=O)NC(=O)C" "c(ccc1)cc1" CC ""
"c(ccc1C(=O)C(C=CC=2)=CC2C(C(=O)O)C)cc1" "c(ccc1C(=O)C(=CC=C2)C=C2)cc1" O=CO C ""
"c(ccc1OC(C=CC=2)=CC2C(C(=O)O)C)cc1" "c(ccc1OC(=CC=C2)C=C2)cc1" O=CO C ""
"O=C(O)C(O)C" O=CO C O ""
"O=C(O)C(O)CO" O=CO OC O ""
"O c(ccc1OC(C(=O)OCC)C)cc1" "c(cc(c1)O)c(c1)O" O=COCC C ""
"c(ccc1C(O)=C2C(C(=CC=C3)C=C3)CC(=O)C)cc1OC2=O" "c(ccc1C(O)=C2)cc1OC2=O"
"c(ccc1)cc1"
"O=C(C)C" ""
"c(ccc1CC(NC(=O)N)C(=O)O)cc1" "c(ccc1C)cc1" "O=C(N)N" O=CO ""
"c(ccc1C=2CC(NC(=O)C)C(=O)O)cc1NC2" "c(ccc1C=2C)cc1NC2" "O=C(N)C" O=CO ""
"O=C(NC(C(=O)O)C(C)C)C" "O=C(N)C" O=CO CCC ""
"Cl-c(ccc1OC(N(C=N2)C=C2)C(=O)C(C)(C)C)cc1" "Cl-c(ccc1O)cc1" "O=CC(C)(C)C"
"N(=CN1)C=C1"
""
"S(C(=C1)C(C(=O)OC)C)C(=C1)C(=O)-c(ccc2)cc2" "S(C=C1)C(=C1)C(=O)-c(ccc2)cc2" O=COC
C ""
"BrC(-c(ccc1)cc1)C" "c(ccc1)cc1" Br C ""
"c(ccc1CC(NC(=O)C)C(=O)O)cc1" "c(ccc1C)cc1" "O=C(N)C" O=CO ""
"c(ccc1C(NC(=O)C)C)cc1" "c(ccc1)cc1" "O=C(N)C" C ""
"c(ccc1C(=O)OC(CCOC(=O)C(=CC=C2)C=C2)C)cc1" "c(ccc1C(=O)OCC)cc1"
"c(ccc1C(=O)O)cc1"
C ""
"c(ccc1C(=O)OC(CCCOC(=O)C(=CC=C2)C=C2)C)cc1" "c(ccc1C(=O)OCCC)cc1"
"c(ccc1C(=O)O)cc1"
C ""
"c(ccc1C(=O)OC(C)CCCCOC(=O)C(=CC=C2)C=C2)cc1" "c(ccc1C(=O)OCCCC)cc1"
"c(ccc1C(=O)O)cc1"
C ""
"c(ccc1C(CCOC(=O)C(=CC=C2)C=C2)C)cc1" "c(ccc1C(=O)OCC)cc1" "c(ccc1)cc1" C ""
"c(ccc1CCC(OC(=O)C(=CC=C2)C=C2)C)cc1" "c(ccc1C(=O)O)cc1" "c(ccc1CC)cc1" C ""
"ClC(=CN=C1O-c(ccc2OC(C(=O)OC)C)cc2)C=C1" "ClC(=CN=C1O-c(ccc2O)cc2)C=C1" O=COC C
""
"FC(=CC=C1O-c(ccc2OC(C(=O)OC)C)cc2)C=C1" "F-c(ccc1OC(=CC=C2O)C=C2)cc1" O=COC C ""
"c(ccc1COC(=O)ONC(C(=O)O)C)cc1" "c(ccc1COC(=O)ON)cc1" O=CO C ""
"c(ccc1COC(=O)ONC(C(=O)O)CCCC)cc1" "c(ccc1COC(=O)ON)cc1" CCCC O=CO ""
"c(ccc1COC(=O)ONC(C(=O)O)C(C)C)cc1" "c(ccc1COC(=O)ON)cc1" CCC O=CO ""
"c(ccc1COC(=O)ONC(C(=O)O)CC(=O)N)cc1" "c(ccc1COC(=O)ON)cc1" "O=C(N)C" O=CO ""
"c(ccc1COC(=O)ONC(C(=O)O)CC(=CC=C2)C=C2)cc1" "c(ccc1COC(=O)ON)cc1" "c(ccc1C)cc1"
O=CO ""
"c(ccc1CC(N)C(=O)N)cc1" "c(ccc1C)cc1" O=CN N ""
"c(ccc1OC(C(=O)O)C)cc1" "c(ccc1O)cc1" O=CO C ""
"c(ccc1C(C(=O)OCC)CC(=CC=C2)C=C2)cc1" "c(ccc1C)cc1" "c(ccc1)cc1" O=COCC ""
"c(ccc1C(=O)OC(C(=CC=C2)C=C2)C)cc1" "c(ccc1C)cc1" "c(ccc1)cc1" C ""
"c(ccc1OC(C(=O)O)C)cc1" "c(ccc1O)cc1" "c(ccc1)cc1" C ""
"O=C(O)C(N)C(O)C" OCC O=CO N ""
"O=C(O)C(N)CO" O=CO OC N ""
"O=C(O)C(N)C" O=CO N C ""
"NC(C)CC" CC C N ""

Fragment Value Knowledge Bases

"" pol 0
"c(ccc1)cc1" pol 1.03565
O=CO pol 0.34022
O pol 0.13185
O=CN pol 0.4066
O=COC pol 0.52369
N pol 0.20746
"c(ccc1C)cc1" pol 1.22046
C pol 0.26192
OC pol 0.32454
NC pol 0.40015
CC pol 0.44496
"c(ccc1CO)cc1" pol 1.28309
"O-C(O)C" pol 0.52327
NCC pol 0.58319
CCCC pol 0.81105
CCC pol 0.62801
"c(cc(c1)C)c(c1)O" pol 1.28555
"c(cc(O)c1O)c(c1)C" pol 1.45115
"c(ccc1C=2C)cc1NC2" pol 1.67509
"O-C(N)CC" pol 0.77955
"c(ccc1NC(=O)O)cc1" pol 1.44699
OCC pol 0.50759
"c(ccc1C(=O)OC)cc1" pol 1.53017
"c(ccc1C(=O)N)cc1" pol 1.41308
"c(ccc1C(=O)O)cc1" pol 1.29877
"OCCN(C)C" pol 1.0312
"N(C=CC=1)=CC1" pol 0.90175
"SC-c(ccc1)cc1" pol 1.51261
"N(=CN1C)C=C1" pol 1.00967
"O=C(OCC)C" pol 0.89664
"c(cc(c1)NC(=O)C)c(c1)OC" pol 1.82324
"N(C)C(C)C" pol 0.95893
"c(ccc1C=O)cc1" pol 1.26804
"c(cc(OC)c1NC2=O)cc1CC2" pol 1.92963
"CN(C)C" pol 0.78553
"O=COCCN(C)C" pol 1.23035
"OC(CC1)CC1" pol 0.97784
"N(=CN1)C=C1C" pol 1.00271
"O=C(OC)C" pol 0.7136
"c(ccc1OC)cc1OC" pol 1.55128
"c(ccc1CC)cc1" pol 1.40351
"c(ccc1C=C2)cc1C=C2" pol 1.75118
"c(ccc1C=C2C=C3)cc1C=C2C=C3" pol 2.46743
"C(H)(H)H" pol 0.21278
"Cl-c(ccc1)cc1" pol 1.22843
"c(ccc1C(C=CC=2)=C(F)C2)cc1" pol 2.03209
"c(ccc1C(=C2)OC)cc1C=C2" pol 2.00798
"N(C=C1)-c(ccc2)c1c2OC" pol 1.7174
"c(cc(c1)CCOC)c(c1)OC" pol 1.9166
"c(ccc1CC=C)cc1OC" pol 1.83243
"O(N=C1)C(=C1)-c(ccc2)cc2" pol 1.78141
"C(CCC1)N(C1)C" pol 1.25579
"CN(C)CC" pol 0.96858
"S(C)CC" pol 0.9298

```
NCCCC pol 0.94928
"O=C(N)NCCC" pol 1.11206
NCCC pol 0.76624
"N=C(N)NCCC" pol 1.21487
"O=C(O)CC" pol 0.70631
"O=C(N)C" pol 0.59651
Br pol 0.361
"CC(C)C" pol 0.81105
"O=CNC(=O)NC(=O)C" pol 1.14148
"c(cc1C(=O)C(=CC=C2)C=C2)cc1" pol 2.25315
"c(cc1OC(=CC=C2)C=C2)cc1" pol 2.0692
"c(cc(c1)O)c(c1)O" pol 1.16589
O=COCC pol 0.70673
"c(ccc1C(O)=C2)cc1OC2=O" pol 1.70371
"O=C(C)C" pol 0.64055
"O=C(N)N" pol 0.55328
"CCN(C(C)C)C(C)C" pol 1.70075
"Cl-c(ccc1O)cc1" pol 1.29379
"O=CC(C)(C)C" pol 1.0006
"N(=CN1)C=C1" pol 0.81698
"S(C=C1)C(=C1)C(=O)-c(ccc2)cc2" pol 2.21242
"c(ccc1C(=O)OCC)cc1" pol 1.71321
"c(ccc1C(=O)OCCC)cc1" pol 1.89625
"c(ccc1C(=O)OCCCC)cc1" pol 2.0793
"ClC(=CN-C1O-c(ccc2O)cc2)C=C1" pol 2.19851
"F-c(ccc1OC(=CC=C2O)C=C2)cc1" pol 2.13556
"c(ccc1COC(=O)ON)cc1" pol 1.70969
"c(ccc1O)cc1" pol 1.10052
C#CC pol 0.53572
C#C pol 0.35268
"FC(F)F" pol 0.26184
"c(ccc1OC=O)cc1" pol 1.29967


"" alpha 0
"c(ccc1)cc1" alpha 0
O=CO alpha 0.5952
O alpha 0.4204
O=CN alpha 0.39068
O=COC alpha 0
N alpha 0.04808
"c(ccc1C)cc1" alpha 0
C alpha 0
OC alpha 0.33647
NC alpha 0.04383
CC alpha 0
"c(ccc1CO)cc1" alpha 0.33154
"O=C(O)C" alpha 0.50165
NCC alpha 0.04187
CCCC alpha 0
CCC alpha 0
"c(cc(c1)C)c(c1)O" alpha 0.63673
"c(cc(O)c1O)c(c1)C" alpha 1.2436
"c(ccc1C=2C)cc1NC2" alpha 0.35449
"O=C(N)CC" alpha 0.38782
"c(ccc1NC(=O)O)cc1" alpha 0.34611
OCC alpha 0.32545
"c(ccc1C(=O)OC)cc1" alpha 0
"c(ccc1C(=O)N)cc1" alpha 0.39297
"c(ccc1C(=O)O)cc1" alpha 0.62298
"OCCN(C)C" alpha 0.35707
"N(C=CC=1)=CC1" alpha 0
"SC=c(ccc1)cc1" alpha 0.09891
"N(=CN1C)C=C1" alpha 0
"O=C(OCC)C" alpha 0.01964
"c(cc(c1)NC(=O)C)c(c1)OC" alpha 0.51371
"N(C)C(C)C" alpha 0.03205
"c(ccc1C=O)cc1" alpha 0
"c(cc(OC)c1NC2=O)cc1OC2" alpha 0.14414
"CN(C)C" alpha 0
"O=COCCN(C)C" alpha 0
"OC(CC1)CC1" alpha 0.30433
"N(=CN1)C=C1C" alpha 0.47626
"O=C(OC)C" alpha 0.01963
"c(ccc1OC)cc1OC" alpha 0
"c(ccc1CC)cc1" alpha 0
"c(ccc1C=C2)cc1C=C2" alpha 0
"c(ccc1C=C2C=C3)cc1C=C2C=C3" alpha 0
"C(H)(H)H" alpha 0
"Cl-c(ccc1)cc1" alpha 0
"c(ccc1C(=CC=2)-C(F)C2)cc1" alpha 0
"c(ccc1C(=C2)OC)cc1C=C2" alpha 0
"N(C=C1)-c(ccc2)c1c2OC" alpha 0.25646
"c(cc(c1)CCOC)c(c1)OC" alpha 0
"c(ccc1CC=C)cc1OC" alpha 0
"O(N=C1)C(=C1)-c(ccc2)cc2" alpha 0
"C(CCC1)N(C1)C" alpha 0
"CN(C)CC" alpha 0
"S(C)CC" alpha 0
NCCCC alpha 0.04056
"O=C(N)NCCC" alpha 0.92846
NCCC alpha 0.04097
"N=C(N)NCCC" alpha 0.76008
"O=C(O)CC" alpha 0.52786
"O=C(N)C" alpha 0.40835
Br alpha 0
"CC(C)C" alpha 0
"O=CNC(=O)NC(=O)C" alpha 1.57597
"c(ccc1C(=O)C(=CC=C2)C=C2)cc1" alpha 0
"c(ccc1OC(=CC=C2)C=C2)cc1" alpha 0
"c(cc(c1)O)c(c1)O" alpha 1.28469
O=COCC alpha 0
"c(ccc1C(O)=C2)cc1OC2=O" alpha 0.75675
"O=C(C)C" alpha 0.03926
"O=C(N)N" alpha 0.94005
"CCN(C(C)C)C(C)C" alpha 0
"Cl-c(ccc1O)cc1" alpha 0.69286
"O=CC(C)(C)C" alpha 0
"N(=CN1)C=C1" alpha 0.47919
"S(C=C1)C(=C1)C(=O)-c(ccc2)cc2" alpha 0
"c(ccc1C(=O)OCC)cc1" alpha 0
"c(ccc1C(=O)OCCC)cc1" alpha 0
"c(ccc1C(=O)OCCCC)cc1" alpha 0
"ClC(=CN=C1O-c(ccc2O)cc2)C=C1" alpha 0.65209
"F-c(ccc1OC(=CC=C2O)C=C2)cc1" alpha 0.65025
"c(ccc1COC(=O)ON)cc1" alpha 0
"c(ccc1O)cc1" alpha 0.65093
C#CC alpha 0
C#C alpha 0
"FC(F)F" alpha 0
"c(ccc1OC=O)cc1" alpha 0


"" beta 0
"c(ccc1)cc1" beta 0.12
O=CO beta 0.4379
O beta 0.36
O=CN beta 0.87117
O=COC beta 0.40072
N beta 0.6621
"c(ccc1C)cc1" beta 0.10002
C beta 0
OC beta 0.41936
NC beta 0.68668
CC beta 0
"c(ccc1CO)cc1" beta 0.53932
"O=C(O)C" beta 0.50001
NCC beta 0.69798
```

```
CCCC beta 0
CCC beta 0
"c(cc(c1)C)c(c1)O" beta 0.24174
"c(cc(O)c1O)c(c1)C" beta 0.48919
"c(ccc1C-2C)cc1NC2" beta 0.46119
"O-C(N)CC" beta 0.97545
"c(ccc1NC(=O)O)cc1" beta 1.15088
OCC beta 0.44667
"c(ccc1C(=O)OC)cc1" beta 0.44884
"c(ccc1C(=O)N)cc1" beta 0.92053
"c(ccc1C(-O)O)cc1" beta 0.42259
"OCCN(C)C" beta 1.10661
"N(C=CC=1)-CC1" beta 0.61995
"SC-c(ccc1)cc1" beta 0.19707
"N(=CN1C)C=C1" beta 0.79798
"O=C(OCC)C" beta 0.506
"c(cc(c1)NC(=O)C)c(c1)OC" beta 1.0514
"N(C)C(C)C" beta 0.73026
"c(ccc1C=O)cc1" beta 0.23716
"c(cc(OC)c1NC2=O)cc1CC2" beta 1.73747
"CN(C)C" beta 0.7313
"O=COCCN(C)C" beta 1.07777
"OC(CC1)CC1" beta 0.49911
"N(=CN1)C=C1C" beta 0.7994
"O-C(OC)C" beta 0.506
"c(ccc1OC)cc1OC" beta 0.39524
"c(ccc1CC)cc1" beta 0.10002
"c(ccc1C=C2)cc1C-C2" beta 0.21
"c(ccc1C=C2C-C3)cc1C=C2C=C3" beta 0.25
"C(H)(H)H" beta 0
"Cl-c(ccc1)cc1" beta 0
"c(ccc1C(C=CC-2)=C(F)C2)cc1" beta 0.10002
"c(ccc1C(=C2)OC)cc1C=C2" beta 0.21673
"N(C=C1)-c(ccc2)c1c2OC" beta 0.72133
"c(cc(c1)CCOC)c(c1)OC" beta 0.47089
"c(ccc1CC=C)cc1OC" beta 0.2759
"O(N=C1)C(=C1)-c(ccc2)cc2" beta 0.11003
"C(CCC1)N(C1)C" beta 0.75777
"CN(C)CC" beta 0.74144
"S(C)CC" beta 0.23476
NCCCC beta 0.70558
"O=C(N)NCCC" beta 1.03954
NCCC beta 0.70319
"N=C(N)NCCC" beta 1.04805
"O=C(O)CC" beta 0.49472
"O-C(N)C" beta 0.95572
Br beta 0
"CC(C)C" beta 0
"O=CNC(=O)NC(=O)C" beta 1.77163
"c(ccc1C(=O)C(=CC=C2)C=C2)cc1" beta 0.31311
"c(ccc1OC(=CC=C2)C=C2)cc1" beta 0.19674
"c(cc(c1)O)c(c1)O" beta 0.44427
O=COCC beta 0.40072
"c(ccc1C(O)=C2)cc1OC2=O" beta 1.19787
"O=C(C)C" beta 0.4776
"O=C(N)N" beta 1.00294
"CCN(C(C)C)C(C)C" beta 0.78188
"Cl-c(ccc1O)cc1" beta 0.17612
"O=CC(C)(C)C" beta 0.29357
"N(=CN1)C=C1" beta 0.77734
"S(C=C1)C(=C1)C(=O)-c(ccc2)cc2" beta 0.35239
"c(ccc1C(=O)OCC)cc1" beta 0.44302
"c(ccc1C(=O)OCCC)cc1" beta 0.44908
"c(ccc1C(=O)OCCCC)cc1" beta 0.4491
"C1C(-CN-C1O-c(ccc2O)cc2)C=C1" beta 0.84945
"F.c(ccc1OC(=CC=C2O)C=C2)cc1" beta 0.33387
"c(ccc1COC(=O)ON)cc1" beta 1.25366
"c(ccc1O)cc1" beta 0.22685
C#CC beta 0
C#C beta 0
"FC(F)F" beta 0.6111
"c(ccc1OC=O)cc1" beta 0.40072


"" vol 0
"c(ccc1)cc1" vol 88.87
O-CO vol 42.69137
O vol 18
O-CN vol 35.62097
"c(ccc1C)cc1" vol 107.35787
C vol 64
OC vol 40.15898
NC vol 49.79824
CC vol 66.9676
"c(ccc1CO)cc1" vol 101.3454
"O=C(O)C" vol 57.5008
NCC vol 67.08871
CCCC vol 99.59081
CCC vol 83.38098
"c(cc(c1)C)c(c1)O" vol 105.08736
"c(cc(O)c1O)c(c1)C" vol 102.44729
"c(ccc1C=2C)cc1NC2" vol 121.69614
"O=C(N)CC" vol 68.97967
"c(ccc1NC(=O)O)cc1" vol 107.31586
OCC vol 58.63084
"c(ccc1C(=O)O)cc1" vol 105.19647
"OCCN(C)C" vol 98.64059
"N(C=CC=1)-CC1" vol 82.15078
"SC-c(ccc1)cc1" vol 116.34286
"c(cc(c1)NC(=O)C)c(c1)OC" vol 141.88159
"N(C)C(C)C" vol 104.99468
"c(ccc1C=O)cc1" vol 104.38103
"c(cc(OC)c1NC2=O)cc1CC2" vol 148.38214
"CN(C)C" vol 91.17316
"OC(CC1)CC1" vol 89.11517
"c(ccc1OC)cc1OC" vol 135.12476
"c(ccc1CC)cc1" vol 123.16602
"c(ccc1C=C2)cc1C=C2" vol 124.03823
"c(ccc1C=C2C=C3)cc1C=C2C=C3" vol 156.73386
"C(H)(H)H" vol 64
"c(ccc1C(=O)N)cc1" vol 92.17992
"c(ccc1C(=O)OC)cc1" vol 125.0551
"O=C(OC)C" vol 79.3998
O=COC vol 61.6403
N vol 26.35
"N(=CN1)C=C1" vol 79.5254
"O=C(OCC)C" vol 97.8785
"O=COCCN(C)C" vol 116.63025
"N(=CN1)C=C1C" vol 75.5254
"Cl-c(ccc1)cc1" vol 102.67341
"c(ccc1C(C=CC=2)=C(F)C2)cc1" vol 161.81456
"c(ccc1C(=C2)OC)cc1C=C2" vol 146.60387
"N(C=C1)-c(ccc2)c1c2OC" vol 127.7012
"c(cc(c1)CCOC)c(c1)OC" vol 167.96375
"c(ccc1CC=C)cc1OC" vol 155.99893
"O(N=C1)C(=C1)-c(ccc2)cc2" vol 123.78
"C(CCC1)N(C1)C" vol 116.89104
"CN(C)CC" vol 106.54556
"S(C)CC" vol 91.99149
NCCCC vol 100.24445
"O-C(N)NCCC" vol 70.48929
NCCC vol 83.8139
"N-C(N)NCCC" vol 140.41
"O-C(O)CC" vol 75.38771
"O-C(N)C" vol 47.11602
Br vol 54.31
"CC(C)C" vol 100.22433
"O=CNC(=O)NC(=O)C" vol 82.27653
```

"c(ccc1C(=O)C(-CC-C2)C-C2)cc1" vol 161.82233
"c(ccc1OC(-CC-C2)C-C2)cc1" vol 154.54443
"c(cc(c1)O)c(c1)O" vol 85.48676
O COCC vol 80.80
"c(ccc1C(O)=C2)cc1OC2=O" vol 129.72
"O-C(C)C" vol 74.6638
"O-C(N)N" vol 58.35143
"CCN(C(C)C)C(C)C" vol 162.97388
"Cl-c(ccc1O)cc1" vol 100.55387
"O=CC(C)(C)C" vol 108.62464
"N(=CN1)C=C1" vol 66.08
"S(C=C1)C(=C1)C(=O)-c(ccc2)cc2" vol 143.21201
"c(ccc1C(=O)OCC)cc1" vol 143.4658
"c(ccc1C(=O)OCCC)cc1" vol 160.5181
"c(ccc1C(=O)OCCCC)cc1" vol 178.23
"ClC(=CN=C1O-c(ccc2O)cc2)C=C1" vol 155.44527
"F-c(ccc1OC(-CC-C2O)C-C2)cc1" vol 161.91039
"c(ccc1OC(-O)ON)cc1" vol 151.97
"c(ccc1O)cc1" vol 88.51344
C#CC vol 60.71
C#C vol 41.95
"FC(F)F" vol 70.68292
"c(ccc1OC=O)cc1" vol 117.43

Column Suggestion Knowledge Base

"c(ccc1C(O)C(=O)O)cc1" "Chiralpak WH" "Daicel p27"
"c(ccc1C(O)C(=O)O)cc1" "Chiralcel OD" "Daicel p27"
"c(ccc1C(O)C(=O)N)cc1" "Chiralcel CA-1" "Daicel p44"
"c(ccc1C(O)C(=O)OC)cc1" "Chiralcel OK" "Daicel p44"
"c(ccc1C(N)C(=O)O)cc1" "Crownpak CR" "Daicel p34"
"c(ccc1CC(N)C(=O)O)cc1" "Crownpak CR" "Daicel p34"
"c(ccc1CC(N)C(=O)O)cc1" "Chiralpak WH" "Daicel p34"
"c(ccc1CC(N)C(=O)O)cc1" "Chiralpak WM" "Daicel p34"
"c(ccc1C(O)C)cc1" "Chiralcel OB" "Daicel p42"
"c(ccc1C(C(=O)O)C)cc1" "Chiralcel OD" "Daicel p27"
"c(ccc1C(C(=O)OC)C)cc1" "Chiralcel OJ" "Daicel p28"
"c(ccc1C(N)CO)cc1" "Crownpak CR" "Daicel p32"
"c(ccc1C(O)CN)cc1" "Crownpak CR" "Daicel p32"
"c(ccc1C(N)C)cc1" "Crownpak CR" "Daicel p32"
"c(ccc1C(O)CC)cc1" "Chiralcel OB" "Daicel p42"
"c(ccc1C(N)C(O)C(-CC-C2)C-C2)cc1" "Crownpak CR" "Daicel p32"
"c(ccc1CC(OC(=O)C)C)cc1" "Chiralcel OB" "Daicel p45"
"NC(C)C(C)C" "Crownpak CR" "Daicel p32"
"OCC(N)CCCC" "Crownpak CR" "Daicel p32"
"OCC(N)CCC" "Crownpak CR" "Daicel p32"
"O-c(ccc1CC(N)C(=O)O)cc1" "Crownpak CR" "Daicel p35"
"O-c(ccc1CC(N)C(=O)O)c(O)c1" "Crownpak CR" "Daicel p35"
"c(ccc1C=2CC(N)C(=O)O)cc1NC2" "Crownpak CR" "Daicel p38"
"c(ccc1C=2CC(N)C(=O)O)cc1NC2" "Chiralpak WH" "Daicel p38"
"O=C(N)CCC(N)C(=O)O" "Crownpak CR" "Daicel p36"
"c(ccc1NC(=O)OC(C)CC)cc1" "Chiralcel OD" "Daicel p46"
"c(ccc1C(C)CO)cc1" "Chiralcel OB" "Daicel p42"
"c(ccc1C(O)CO)cc1" "Chiralcel OB" "Daicel p44"
"c(ccc1C(O)CCO)cc1" "Chiralcel OB" "Daicel p44"
"c(ccc1C(=O)OCC(NC(=O)C(=CC=C2)C=C2)CC)cc1" "Chiralcel OB" "Daicel p46"
"c(ccc1CC(OC(=O)C(=CC=C2)C=C2)C)cc1" "Chiralcel OB" "Daicel p45"
"c(cc(c1)C(OCCN(C)C)C(N=CC=2)=CC2)c(c1)H" "Chiralcel OD" "Daicel p21"
"O-C(OCCC(OC(=O)C)C)C" "Chiralcel OB" "Daicel p45"
"c(cc(c1)OCC(O)CNC(C)c(c1)NC(=O)C" "Chiralcel OD" "Daicel p20"
"c(ccc1C(=O)C(O)C(-CC=C2)C=C2)cc1" "Chiralcel OD" "Daicel p44"
"c(ccc1C(=O)C(O)C(-CC=C2)C=C2)cc1" "Chiralcel OF" "Daicel p44"
"c(ccc1C(=O)C(O)C(=CC=C2)C=C2)cc1" "Chiralpak OP" "Daicel p44"
"O=C(N=c(c1cc2)c(c2)OCC(O)CN(C)C)CC1" "Chiralcel OD" "Daicel p20"
"c(ccc1C(C(=O)OCCN(C)C)C(CC2)(CC2)O)cc1" "Chiralcel OJ" "Daicel p21"
"c(ccc1C(C(=O)OCCN(C)C)C(CC2)(CC2)O)cc1" "Chiralcel OD" "Daicel p21"
"N(=CN1)C=C1CC(N)C(=O)O" "Crownpak CR" "Daicel p38"
"N(=CN1)C=C1CC(N)C(=O)O" "Chiralpak WH" "Daicel p38"
"c(cc(c1)C(CN)CC(=O)O)c(c1)H" "Crownpak CR" "Daicel p35"
"c(ccc1C(=O)OC(C)CC(=O)OC)cc1" "Chiralcel OB" "Daicel p45"
"c(ccc1OCC(O)CO)cc1OC" "Chiralcel OD" "Daicel p44"
"c(ccc1C(=C2)C(O)C)cc1C=C2" "Chiralcel OB" "Daicel p43"
"c(ccc1C(=C2)C(O)C)cc1C=C2" "Chiralcel OC" "Daicel p43"
"c(ccc1C(=C2)C(O)C)cc1C=C2" "Chiralcel OJ" "Daicel p43"
"c(ccc1C(=C2)C(O)C)cc1C=C2" "Chiralcel OD" "Daicel p43"
"c(ccc1C-C2C(O)C)cc1C-C2" "Chiralcel OJ" "Daicel p43"
"c(ccc1C-C2C(O)C)cc1C-C2" "Chiralcel OB" "Daicel p43"
"c(ccc1C=C2C(O)C)cc1C=C2" "Chiralcel OK" "Daicel p43"
"c(ccc1C=C2C-C3)cc1C(=C2C=C3)C(O)C(H)(H)H" "Chiralcel OD" "Daicel p43"
"c(ccc1C=C2C=C3)cc1C(=C2C=C3)C(O)C(H)(H)H" "Chiralcel OG" "Daicel p43"
"c(ccc1C=C2C=C3)cc1C(=C2C=C3)C(O)C(H)(H)H" "Chiralpak OT" "Daicel p43"
"c(ccc1C=C2C-C3)cc1C(=C2C=C3)C(O)C(H)(H)H" "Chiralcel CA-1" "Daicel p43"
"Cl-c(ccc1C(OCCN(C)C)C(N=CC=2)=CC2)cc1" "Chiralcel OD" "Daicel p21"
"Cl-c(ccc1C(=N)CC(=O)O)cc1" "Crownpak CR" "Daicel p35"
"Cl-c(cc(c1)c1C(SCC(=CC=C2)C=C2)CN(C=N4)C=C4)cc1" "Chiralcel OD" "Daicel p22"
"Cl-c(cc(c1)c1C(SCC(-CC=C2)C=C2)CN(C=N4)C=C4)cc1" "Chiralcel OJ" "Daicel p22"
"c(ccc1C(O)CC(=CC=C2)C=C2)cc1" "Chiralcel OD" "Daicel p46"
"C(=CC=C1)-c(ccc2)c1c2OCC(O)CNC(C)C" "Chiralcel OD" "Daicel p20"
"N(C=C1)-c(ccc2)c1c2OCC(O)CNC(C)C" "Chiralcel OD" "Daicel p20"
"c(cc(c1)OCC(O)CNC(C)C)c(c1)CCOC" "Chiralcel OD" "Daicel p20"
"c(ccc1OCC(O)CNC(C)C)cc1CC=C" "Chiralcel OD" "Daicel p20"
"c(ccc1C(ON=2)=CC2C(O)CN(CCC3)CC3)cc1" "Chiralcel OD" "Daicel p20"
"c(ccc1C(ON-2)=CC2C(O)CN(CCC3)CC3)cc1" "Chiralcel OF" "Daicel p20"
"S(C)CCC(N)C(-O)O" "Chiralpak WM" "Daicel p36"
"S(C)CCC(N)C(-O)O" "Crownpak CR" "Daicel p36"
"S(C)CCC(N)C(-O)O" "Chiralpak WH" "Daicel p36"
"S(C)CCC(N)C(-O)O" "Chiralpak WE" "Daicel p36"
"O-C(O)C(N)CCCCN" "Crownpak CR" "Daicel p38"
"O-C(O)C(N)CCCCN" "Chiralpak WM" "Daicel p38"
"O-C(O)C(N)CCCCN" "Chiralpak WE" "Daicel p38"
"O=C(N)NCCCC(N)C(=O)O" "Crownpak CR" "Daicel p37"
"O-C(O)C(N)C(C)C" "Crownpak CR" "Daicel p33"
"O-C(O)C(N)C(C)C" "Chiralpak WH" "Daicel p33"
"O-C(O)C(N)C(C)C" "Chiralpak WM" "Daicel p33"
"O-C(O)C(N)C(C)C" "Chiralpak WE" "Daicel p33"
"O-C(O)C(N)C(C)CC" "Crownpak CR" "Daicel p34"
"O-C(O)C(N)C(C)CC" "Chiralpak WH" "Daicel p34"
"O-C(O)C(N)C(C)CC" "Chiralpak WM" "Daicel p34"
"O-C(O)C(N)C(C)CC" "Chiralpak WE" "Daicel p34"
"O-C(O)C(N)CCCN" "Crownpak CR" "Daicel p38"
"O-C(O)C(N)CCCN" "Chiralpak WM" "Daicel p38"
"O=C(O)C(N)CCCNC(=N)N" "Crownpak CR" "Daicel p39"
"O=C(O)C(N)CCCNC(=N)N" "Chiralpak WE" "Daicel p39"
"O=C(O)C(N)CCCNC(=N)N" "Chiralpak WM" "Daicel p39"
"O-C(O)CCC(N)C(-O)O" "Crownpak CR" "Daicel p36"
"O=C(N)CC(N)C(=O)O" "Crownpak CR" "Daicel p37"
"O=C(O)CC(N)C(=O)O" "Crownpak CR" "Daicel p37"
"O=C(O)CC(N)C(=O)O" "Chiralpak WH" "Daicel p37"
"O=C(O)CC(N)C(=O)O" "Chiralpak WM" "Daicel p37"
"O=C(O)C(N)CC(C)C" "Chiralpak WE" "Daicel p33"
"O-C(O)C(N)CC(C)C" "Chiralpak WM" "Daicel p33"
"c(ccc1C(C(-O)NC(=O)NC(=O)C)CC)cc1" "Chiralcel OF" "Daicel p28"
"c(ccc1C(C(-O)NC(=O)NC(=O)C)CC)cc1" "Chiralcel OG" "Daicel p28"
"c(ccc1C(=O)C(C=CC=2)=CC2C(C(=O)O)C)cc1" "Chiralcel OJ" "Daicel p27"
"c(ccc1OC(C=CC=2)=CC2C(C(=O)O)C)cc1" "Chiralcel OJ" "Daicel p27"
"O=C(O)C(O)C" "Chiralpak WH" "Daicel p27"
"O=C(O)C(O)CO" "Chiralpak WH" "Daicel p27"
"O-c(ccc1OC(C(-O)OCC)C)cc1" "Chiralcel OJ" "Daicel p29"
"c(ccc1C(O)-c2C(C(-CC-C3)C-C3)CC(-O)C)cc1OC2=O" "Chiralpak WH" "Daicel p49"
"c(ccc1CC(NC(-O)N)C(-O)O)cc1" "Chiralpak WH" "Daicel p41"
"c(ccc1C-2CC(NC(-O)C)C(=O)O)cc1NC2" "Chiralpak WH" "Daicel p49"
"O-C(NC(C(=O)O)C(C)C)C" "Chiralpak WH" "Daicel p49"
"Cl-c(ccc1OC(N(C=N2)C=C2)C(=O)C(C)(C)C)cc1" "Chiralcel OJ" "Daicel p22"
"S(C(=C1)C(C(=O)OC)C)C(=C1)C(=O)-c(ccc2)cc2" "Chiralcel OJ" "Daicel p28"
"BrC(-c(ccc1)cc1)C" "Chiralpak OT" "Daicel p46"
"c(ccc1CC(NC(-O)C)C(=O)O)cc1" "Chiralpak WH" "Daicel p39"
"c(ccc1C(NC(=O)C)C)cc1" "Chiralcel OB" "Daicel p46"

"c(ccc1C(=O)OC(CCOC(=O)C(-CC=C2)C=C2)C)cc1" "Chiralcel OR" "Daicel p45"
"c(ccc1C(=O)OC(CCCOC(=O)C(-CC=C2)C=C2)C)cc1" "Chiralcel OR" "Daicel p45"
"c(ccc1C(=O)OC(C)CCCCOC(=O)C(=CC=C2)C=C2)cc1" "Chiralcel OR" "Daicel p45"
"c(ccc1C(CCOC(=O)C(=CC=C2)C=C2)C)cc1" "Chiralcel OR" "Daicel p45"
"c(ccc1CCC(OC(=O)C(-CC=C2)C=C2)C)cc1" "Chiralcel OR" "Daicel p45"
"ClC(=CN=C1O-c(ccc2OC(C(=O)OC)C)cc2)C=C1" "Chiralcel OR" "Daicel p29"
"FC(=CC=C1O-c(ccc2OC(C(=O)OC)C)cc2)C=C1" "Chiralcel OR" "Daicel p29"
"c(ccc1COC(=O)ONC(C(=O)O)C)cc1" "Chiralpak WH" "Daicel p40"
"c(ccc1COC(=O)ONC(C(=O)O)C)cc1" "Chiralcel OD" "Daicel p40"
"c(ccc1COC(=O)ONC(C(=O)O)CCCC)cc1" "Chiralpak WH" "Daicel p40"
"c(ccc1COC(=O)ONC(C(=O)O)CCCC)cc1" "Chiralcel OD" "Daicel p40"
"c(ccc1COC(=O)ONC(C(=O)O)C(C)C)cc1" "Chiralpak WH" "Daicel p40"
"c(ccc1COC(=O)ONC(C(=O)O)C(C)C)cc1" "Chiralcel OD" "Daicel p40"
"c(ccc1COC(=O)ONC(C(=O)O)CC(=O)N)cc1" "Chiralpak WH" "Daicel p40"
"c(ccc1COC(=O)ONC(C(=O)O)CC(=O)N)cc1" "Chiralcel OD" "Daicel p40"
"c(ccc1COC(=O)ONC(C(=O)O)CC(=CC=C2)C=C2)cc1" "Chiralpak WH" "Daicel p40"
"c(ccc1CC(N)C(=O)N)cc1" "Chiralpak WH" "Daicel p41"
"c(ccc1OC(C(=O)O)C)cc1" "Chiralcel OD" "Daicel p29"
"c(ccc1OC(C(=O)O)C)cc1" "Chiralcel OF" "Daicel p26"
"c(ccc1C(C(=O)OCC)CC(=CC=C2)C=C2)cc1" "Chiralpak OT" "Daicel p46"
"c(ccc1C(=O)OC(C(=CC=C2)C=C2)C)cc1" "Chiralpak OT" "Daicel p46"
"c(ccc1C(=O)OC(C)CC#C)cc1" "Chiralcel OR" "Daicel p45"
"c(ccc1C(=O)OC(C#C)C)cc1" "Chiralcel OR" "Daicel p45"
"c(ccc1OC(C(=O)OC)C)cc1" "Chiralcel OJ" "Daicel p29"
"c(ccc1OC(C(=O)OC)C)cc1" "Chiralcel OK" "Daicel p29"
"O=C(O)C(N)C(O)C" "Crownpak CR" "Daicel p35"
"O=C(O)C(N)C(O)C" "Chiralpak WH" "Daicel p35"
"O=C(O)C(N)CO" "Crownpak CR" "Daicel p35"
"O=C(O)C(N)CO" "Chiralpak WH" "Daicel p35"
"O=C(O)C(N)C" "Crownpak CR" "Daicel p33"
"O=C(O)C(N)C" "Chiralpak WH" "Daicel p33"
"O=C(O)C(N)C" "Chiralpak WM" "Daicel p33"
"NC(C)CC" "Crownpak CR" "Daicel p32"

APPENDIX H: OPTIMER Testing Results

Appendix H contains the dialog of the testing summarized in section VII.B.1. It is a log file of the testing procedure.

```
OPTIMEK4.LOG started  14 AUG 1994 13:43:29
   YES
   YES
   YES
   YES
   YES
   YES
   YES
   YES
   YES
   YES


What is the name of the rule base you want to run tests on?
      Please enter it here -> chiral-b


The Rulebase ->chiral-b<- has been loaded.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->MANDELAMIDE<- )

Looking for a C=O in MANDELAMIDE:
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
*    Found C=O bond in MANDELAMIDE at atom 9.
*   For carbonyl, Found that CdO exists. [MANDELAMIDE]

*    Found Benzene (Benz) in MANDELAMIDE at atom 1: ring - (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [MANDELAMIDE]

Looking for a -OH in MANDELAMIDE:
  Found Oxygen at atom 8.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in MANDELAMIDE at atom 8.
*   For hydroxyl, Found that OsH exists. [MANDELAMIDE]

Looking for an Amine in MANDELAMIDE:
*    Found Amine (-NR2) in MANDELAMIDE at atom 11.
*   For amine, Found that N3 exists. [MANDELAMIDE]

Looking for a Primary Amine (NRH2) in MANDELAMIDE:
*    Found Primary Amine (-NH2) in MANDELAMIDE at atom 11.
*   For primary-amine, Found that NP exists. [MANDELAMIDE]

Looking for a Carboxylic acid (XC=OOH) in MANDELAMIDE:
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
Looking for a Carboxylic acid (XC=OOH) in MANDELAMIDE:
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
              (molecule name is ->MANDELAMIDE<- )
p-4   WITH A PROBABILITY OF   83
p-5-7   WITH A PROBABILITY OF   75
p-10-11   WITH A PROBABILITY OF   55
p-12   WITH A PROBABILITY OF   35
p-8-9   WITH A PROBABILITY OF   35
p-14   WITH A PROBABILITY OF   20
p-16   WITH A PROBABILITY OF   6
p-13   WITH A PROBABILITY OF   -53
p-15   WITH A PROBABILITY OF   -82

The following is the data for (choose-a-chiral : A A A A A B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose no-COOH.
***** MANDELAMIDE is found on Page 4.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->PRACTOLOL<- )

Looking for a C=O in PRACTOLOL:
  Found Oxygen at atom 9.  Is it a C=O?
*    Found C=O bond in PRACTOLOL at atom 8.
*   For carbonyl, Found that CdO exists. [PRACTOLOL]

*    Found Benzene (Benz) in PRACTOLOL at atom 1: ring - (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [PRACTOLOL]

Looking for a -OH in PRACTOLOL:
  Found Oxygen at atom 9.
  Found Oxygen at atom 11.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 15.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in PRACTOLOL at atom 15.
*   For hydroxyl, Found that OsH exists. [PRACTOLOL]

Looking for an Amine in PRACTOLOL:
*    Found Amine (-NR2) in PRACTOLOL at atom 7.
*   For amine, Found that N3 exists. [PRACTOLOL]

Looking for a Primary Amine (NRH2) in PRACTOLOL:
Looking for a Primary Amine (NRH2) in PRACTOLOL:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in PRACTOLOL:
  Found Oxygen at atom 9.  Is it a C=O?
  Have C=O at atom 8.  Is there a C-O bond?
  Found Oxygen at atom 11.  Is it a C=O?
  Found Oxygen at atom 15.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in PRACTOLOL:
  Found Oxygen at atom 9.  Is it a C=O?
  Have C=O at atom 8.  Is there a C-O bond?
  Found Oxygen at atom 11.  Is it a C=O?
  Found Oxygen at atom 15.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
              (molecule name is ->PRACTOLOL<- )
p-4   WITH A PROBABILITY OF   83
p-5-7   WITH A PROBABILITY OF   75
p-10-11   WITH A PROBABILITY OF   55
p-16   WITH A PROBABILITY OF   55
p-12   WITH A PROBABILITY OF   35
```

```
p-8-9   WITH A PROBABILITY OF    35
p-15    WITH A PROBABILITY OF    16
p-13    WITH A PROBABILITY OF    53
p-14    WITH A PROBABILITY OF    68

The following is the data for (choose-a-chiral : A A A A A B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** PRACTOLOL is found on Page 4.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function grou.
       (molecule name is ->2 BUTYL PHENYL CARBAMATE<- )

Looking for a C=O in 2 BUTYL PHENYL CARBAMATE:
  Found Oxygen at atom 9.  Is it a C=O?
*    Found C=O bond in 2 BUTYL PHENYL CARBAMATE at atom 8.
*   For carbonyl, Found that CdO exists. [2 BUTYL PHENYL CARBAMATE]

*    Found Benzene (Benz) in 2 BUTYL PHENYL CARBAMATE at atom 1: ring -
   (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [2 BUTYL PHENYL CARBAMATE]

Looking for a -OH in 2 BUTYL PHENYL CARBAMATE:
  Found Oxygen at atom 9.
  Found Oxygen at atom 10.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
Looking for a -OH in 2 BUTYL PHENYL CARBAMATE:
  Found Oxygen at atom 9.
  Found Oxygen at atom 10.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in 2 BUTYL PHENYL CARBAMATE:
*    Found Amine (-NR2) in 2 BUTYL PHENYL CARBAMATE at atom 7.
*   For amine, Found that N3 exists. [2 BUTYL PHENYL CARBAMATE]

Looking for a Primary Amine (NRH2) in 2 BUTYL PHENYL CARBAMATE:
Looking for a Primary Amine (NRH2) in 2 BUTYL PHENYL CARBAMATE:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in 2 BUTYL PHENYL CARBAMATE:
  Found Oxygen at atom 9.  Is it a C=O?
  Have C=O at atom 8.  Is there a C-O bond?
  Have C-O at atom 10.  Is there a -OH bond?
  Found Oxygen at atom 10.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in 2 BUTYL PHENYL CARBAMATE:
  Found Oxygen at atom 9.  Is it a C=O?
  Have C=O at atom 8.  Is there a C-O bond?
  Have C-O at atom 10.  Is there a -OH bond?
  Found Oxygen at atom 10.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->2 BUTYL PHENYL CARBAMATE<- )
p-5-7   WITH A PROBABILITY OF    91
p-12    WITH A PROBABILITY OF    75
```



```
p-8-9   WITH A PROBABILITY OF    75
p-16    WITH A PROBABILITY OF    55
p-4    WITH A PROBABILITY OF    55
p-13    WITH A PROBABILITY OF    35
p-10-11    WITH A PROBABILITY OF    -16
p-15    WITH A PROBABILITY OF    -16
p-14    WITH A PROBABILITY OF    -68

The following is the data for (choose-a-chiral : A A B A B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** 2 BUTYL PHENYL CARBAMATE is found on Page 4.


We are currently looking at choose-a-chiral, trying to decide on
     which a column to choose based on function grou.
       (molecule name is ->D-PHENYLALANINE<- )

Looking for a C=O in D-PHENYLALANINE:
  Found Oxygen at atom 11.  Is it a C=O?
  Found Oxygen at atom 12.  Is it a C=O?
*    Found C=O bond in D-PHENYLALANINE at atom 8.
*   For carbonyl, Found that CdO exists. [D-PHENYLALANINE]

*    Found Benzene (Benz) in D-PHENYLALANINE at atom 1: ring - (1 3 6 10 7 4).
*   For aromatic-ring, Found that has-benz exists. [D-PHENYLALANINE]

Looking for a -OH in D-PHENYLALANINE:
  Found Oxygen at atom 11.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 12.
Looking for a -OH in D-PHENYLALANINE:
  Found Oxygen at atom 11.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 12.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in D-PHENYLALANINE:
*    Found Amine (-NR2) in D-PHENYLALANINE at atom 9.
*   For amine, Found that N3 exists. [D PHENYLALANINE]

Looking for a Primary Amine (NRH2) in D-PHENYLALANINE:
*    Found Primary Amine (-NH2) in D-PHENYLALANINE at atom 9.
*   For primary-amine, Found that NP exists. [D-PHENYLALANINE]

Looking for a Carboxylic acid (XC=OOH) in D-PHENYLALANINE:
  Found Oxygen at atom 11.  Is it a C=O?
  Found Oxygen at atom 12.  Is it a C=O?
  Have C=O at atom 8.  Is there a C-O bond?
  Have C-O at atom 11.  Is there a -OH bond?
*    Found Carboxylic Acid (COOH) in D-PHENYLALANINE at atom 8.
*   For carboxylic-acid, Found that COOH exists. [D-PHENYLALANINE]

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->D-PHENYLALANINE<- )
p-5-7   WITH A PROBABILITY OF    91
p-12    WITH A PROBABILITY OF    75
p-8-9   WITH A PROBABILITY OF    75
p-4    WITH A PROBABILITY OF    55
p-13    WITH A PROBABILITY OF    35
```

```
p-14   WITH A PROBABILITY OF   20
p-15   WITH A PROBABILITY OF   6
p-10-11   WITH A PROBABILITY OF   -16
p-16   WITH A PROBABILITY OF   82

The following is the data for (choose-a-chiral : A A B A A A).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose COOH.
***** D-PHENYLALANINE is found on Page 5.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
       (molecule name is ->HISTIDINE<- )

Looking for a C=O in HISTIDINE:
  Found Oxygen at atom 10.  Is it a C=O?
*    Found C-O bond in HISTIDINE at atom 9.
*   For carbonyl, Found that CdO exists. [HISTIDINE]

*   For aromatic-ring, Found has-benz dosn't exist, so it's no-benz.

Looking for a -OH in HISTIDINE:
  Found Oxygen at atom 10.
  Found Oxygen at atom 11.
  Single bond w/ 'O' found.  Look for bond w/another atom.
Looking for a -OH in HISTIDINE:
  Found Oxygen at atom 10.
  Found Oxygen at atom 11.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in HISTIDINE:
*    Found Amine (-NR2) in HISTIDINE at atom 5.
*   For amine, Found that N3 exists. [HISTIDINE]

Looking for a Primary Amine (NRH2) in HISTIDINE:
*    Found Primary Amine (-NH2) in HISTIDINE at atom 8.
*   For primary-amine, Found that NP exists. [HISTIDINE]

Looking for a Carboxylic acid (XC=OOH) in HISTIDINE:
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
  Have C-O at atom 11.  Is there a -OH bond?
*    Found Carboxylic Acid (COOH) in HISTIDINE at atom 9.
*   For carboxylic-acid, Found that COOH exists. [HISTIDINE]

Finished asking questions!!

End of Rules!

We are currently looking at choose-a chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->HISTIDINE<- )
p-5-7   WITH A PROBABILITY OF   75
p-14   WITH A PROBABILITY OF   69
p-15   WITH A PROBABILITY OF   55
p-12   WITH A PROBABILITY OF   48
p-8-9   WITH A PROBABILITY OF   35
p-16   WITH A PROBABILITY OF   -16
p-4   WITH A PROBABILITY OF   -16
p-13   WITH A PROBABILITY OF   -35
p-10-11   WITH A PROBABILITY OF   -82

The following is the data for (choose-a-chiral : A B B A A A).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose no-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose COOH.
***** HISTIDINE is found on Page 5.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->2-AMINO-3-PHENYLPROPIONAMIDE<- )

Looking for a C=O in 2-AMINO-3-PHENYLPROPIONAMIDE:
  Found Oxygen at atom 5.  Is it a C=O?
*    Found C=O bond in 2-AMINO-3-PHENYLPROPIONAMIDE at atom 4.
*   For carbonyl, Found that CdO exists. [2-AMINO-3-PHENYLPROPIONAMIDE]

*    Found Benzene (Benz) in 2-AMINO-3-PHENYLPROPIONAMIDE at atom 6: ring -
   (6 7 8 9 10 11).
*   For aromatic-ring, Found that has-benz exists. [2-AMINO-3-PHENYLPROPIONAMIDE]

Looking for a -OH in 2-AMINO-3-PHENYLPROPIONAMIDE:
  Found Oxygen at atom 5.
Looking for a -OH in 2-AMINO-3-PHENYLPROPIONAMIDE:
  Found Oxygen at atom 5.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in 2-AMINO-3-PHENYLPROPIONAMIDE:
*    Found Amine (-NR2) in 2-AMINO-3-PHENYLPROPIONAMIDE at atom 1.
*   For amine, Found that N3 exists. [2-AMINO-3-PHENYLPROPIONAMIDE]

Looking for a Primary Amine (NRH2) in 2-AMINO-3-PHENYLPROPIONAMIDE:
*    Found Primary Amine (-NH2) in 2-AMINO-3-PHENYLPROPIONAMIDE at atom 1.
*   For primary-amine, Found that NP exists. [2-AMINO-3-PHENYLPROPIONAMIDE]

Looking for a Carboxylic acid (XC-OOH) in 2-AMINO-3-PHENYLPROPIONAMIDE:
  Found Oxygen at atom 5.  Is it a C=O?
  Have C=O at atom 4.  Is there a C-O bond?
Looking for a Carboxylic acid (XC=OOH) in 2-AMINO-3-PHENYLPROPIONAMIDE:
  Found Oxygen at atom 5.  Is it a C-O?
  Have C=O at atom 4.  Is there a C-O bond?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->2-AMINO-3-PHENYLPROPIONAMIDE<- )
p-5-7   WITH A PROBABILITY OF   91
p-12   WITH A PROBABILITY OF   75
p-8-9   WITH A PROBABILITY OF   75
p-4   WITH A PROBABILITY OF   55
p-13   WITH A PROBABILITY OF   35
p-14   WITH A PROBABILITY OF   20
p-16   WITH A PROBABILITY OF   6
p-10-11   WITH A PROBABILITY OF   -16
p-15   WITH A PROBABILITY OF   -82

The following is the data for (choose-a-chiral : A A B A A B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose no-COOH.
***** 2-AMINO-3-PHENYLPROPIONAMIDE is found on Page 5.
```

```
We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->TRIADIMEFON<- )

Looking for a C=O in TRIADIMEFON:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 17.  Is it a C=O?
*    Found C=O bond in TRIADIMEFON at atom 15.
*   For carbonyl, Found that CdO exists. [TRIADIMEFON]

*    Found Benzene (Benz) in TRIADIMEFON at atom 8: ring - (8 9 10 11 12 13).
*   For aromatic-ring, Found that has-benz exists. [TRIADIMEFON]

Looking for a -OH in TRIADIMEFON:
  Found Oxygen at atom 7.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 17.
Looking for a -OH in TRIADIMEFON:
  Found Oxygen at atom 7.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 17.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in TRIADIMEFON:
*    Found Amine (-NR2) in TRIADIMEFON at atom 2.
*   For amine, Found that N3 exists. [TRIADIMEFON]

Looking for a Primary Amine (NRH2) in TRIADIMEFON:
Looking for a Primary Amine (NRH2) in TRIADIMEFON:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in TRIADIMEFON:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 17.  Is it a C=O?
  Have C=O at atom 15.  Is there a C-O bond?
Looking for a Carboxylic acid (XC=OOH) in TRIADIMEFON:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 17.  Is it a C=O?
  Have C=O at atom 15.  Is there a C-O bond?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
            (molecule name is ->TRIADIMEFON<- )
p-5-7   WITH A PROBABILITY OF   91
p-12   WITH A PROBABILITY OF   75
p-8-9   WITH A PROBABILITY OF   75
p-16   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-13   WITH A PROBABILITY OF   35
p-10-11   WITH A PROBABILITY OF   -16
p-15   WITH A PROBABILITY OF   -16
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : A A B A B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** TRIADIMEFON is found on Page 6.
```



```
We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->ACETYLPHENETHOLIDE<- )

Looking for a C=O in ACETYLPHENETHOLIDE:
  Found Oxygen at atom 11.  Is it a C=O?
*    Found C=O bond in ACETYLPHENETHOLIDE at atom 10.
*   For carbonyl, Found that CdO exists. [ACETYLPHENETHOLIDE]

*    Found Benzene (Benz) in ACETYLPHENETHOLIDE at atom 1: ring - (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [ACETYLPHENETHOLIDE]

Looking for a -OH in ACETYLPHENETHOLIDE:
  Found Oxygen at atom 11.
  Found Oxygen at atom 14.
  Found Oxygen at atom 17.
Looking for a -OH in ACETYLPHENETHOLIDE:
  Found Oxygen at atom 11.
  Found Oxygen at atom 14.
  Found Oxygen at atom 17.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in ACETYLPHENETHOLIDE:
*    Found Amine (-NR2) in ACETYLPHENETHOLIDE at atom 12.
*   For amine, Found that N3 exists. [ACETYLPHENETHOLIDE]

Looking for a Primary Amine (NRH2) in ACETYLPHENETHOLIDE:
Looking for a Primary Amine (NRH2) in ACETYLPHENETHOLIDE:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in ACETYLPHENETHOLIDE:
  Found Oxygen at atom 11.  Is it a C=O?
  Have C=O at atom 10.  Is there a C-O bond?
  Found Oxygen at atom 14.  Is it a C=O?
  Have C=O at atom 13.  Is there a C-O bond?
  Found Oxygen at atom 17.  Is it a C=O?
  Have C=O at atom 16.  Is there a C-O bond?
Looking for a Carboxylic acid (XC=OOH) in ACETYLPHENETHOLIDE:
  Found Oxygen at atom 11.  Is it a C=O?
  Have C=O at atom 10.  Is there a C-O bond?
  Found Oxygen at atom 14.  Is it a C=O?
  Have C=O at atom 13.  Is there a C-O bond?
  Found Oxygen at atom 17.  Is it a C=O?
  Have C=O at atom 16.  Is there a C-O bond?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
            (molecule name is ->ACETYLPHENETHOLIDE<- )
p-5-7   WITH A PROBABILITY OF   91
p-12   WITH A PROBABILITY OF   75
p-8-9   WITH A PROBABILITY OF   75
p-16   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-13   WITH A PROBABILITY OF   35
p-10-11   WITH A PROBABILITY OF   -16
p-15   WITH A PROBABILITY OF   -16
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : A A B A B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
```

```
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** ACETYLPHENETHOLIDE is found on Page 6.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->N-(PHENYLETHYL)ACETAMIDE<- )

Looking for a C=O in N-(PHENYLETHYL)ACETAMIDE:
  Found Oxygen at atom 10.  Is it a C=O?
*    Found C=O bond in N-(PHENYLETHYL)ACETAMIDE at atom 9.
*   For carbonyl, Found that CdO exists. [N-(PHENYLETHYL)ACETAMIDE]

*    Found Benzene (Benz) in N-(PHENYLETHYL)ACETAMIDE at atom 1: ring -
   (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [N-(PHENYLETHYL)ACETAMIDE]

Looking for a -OH in N-(PHENYLETHYL)ACETAMIDE:
  Found Oxygen at atom 10.
Looking for a -OH in N-(PHENYLETHYL)ACETAMIDE:
  Found Oxygen at atom 10.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in N-(PHENYLETHYL)ACETAMIDE:
*    Found Amine (-NR2) in N-(PHENYLETHYL)ACETAMIDE at atom 11.
*   For amine, Found that N3 exists. [N-(PHENYLETHYL)ACETAMIDE]

Looking for a Primary Amine (NRH2) in N-(PHENYLETHYL)ACETAMIDE:
Looking for a Primary Amine (NRH2) in N-(PHENYLETHYL)ACETAMIDE:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in N-(PHENYLETHYL)ACETAMIDE:
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
Looking for a Carboxylic acid (XC=OOH) in N-(PHENYLETHYL)ACETAMIDE:
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->N-(PHENYLETHYL)ACETAMIDE<- )
p-5-7   WITH A PROBABILITY OF   91
p-12   WITH A PROBABILITY OF   75
p-8-9   WITH A PROBABILITY OF   75
p-16   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-13   WITH A PROBABILITY OF   35
p-10-11   WITH A PROBABILITY OF   -16
p-15   WITH A PROBABILITY OF   -16
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : A A B A B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** N-(PHENYLETHYL)ACETAMIDE is found on Page 6.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->N,O-DIBENZOYL2-AMINOBUTANOL<- )

Looking for a C=O in N,O-DIBENZOYL2-AMINOBUTANOL:
  Found Oxygen at atom 6.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
*    Found C=O bond in N,O-DIBENZOYL2-AMINOBUTANOL at atom 7.
*   For carbonyl, Found that CdO exists. [N,O-DIBENZOYL2-AMINOBUTANOL]

*    Found Benzene (Benz) in N,O-DIBENZOYL2-AMINOBUTANOL at atom 9: ring -
   (9 10 11 12 13 14).
*   For aromatic ring, Found that has-benz exists. [N,O-DIBENZOYL2-AMINOBUTANOL]

Looking for a -OH in N,O-DIBENZOYL2-AMINOBUTANOL:
  Found Oxygen at atom 6.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 8.
  Found Oxygen at atom 16.
Looking for a -OH in N,O-DIBENZOYL2-AMINOBUTANOL:
  Found Oxygen at atom 6.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 8.
  Found Oxygen at atom 16.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in N,O-DIBENZOYL2-AMINOBUTANOL:
*    Found Amine (-NR2) in N,O-DIBENZOYL2-AMINOBUTANOL at atom 2.
*   For amine, Found that N3 exists. [N,O-DIBENZOYL2-AMINOBUTANOL]

Looking for a Primary Amine (NRH2) in N,O-DIBENZOYL2-AMINOBUTANOL:
Looking for a Primary Amine (NRH2) in N,O-DIBENZOYL2-AMINOBUTANOL:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in N,O-DIBENZOYL2-AMINOBUTANOL:
  Found Oxygen at atom 6.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
  Have C=O at atom 7.  Is there a C-O bond?
  Have C-O at atom 6.  Is there a -OH bond?
  Found Oxygen at atom 16.  Is it a C=O?
  Have C=O at atom 15.  Is there a C-O bond?
Looking for a Carboxylic acid (XC=OOH) in N,O-DIBENZOYL2-AMINOBUTANOL:
  Found Oxygen at atom 6.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
  Have C=O at atom 7.  Is there a C-O bond?
  Have C-O at atom 6.  Is there a -OH bond?
  Found Oxygen at atom 16.  Is it a C=O?
  Have C=O at atom 15.  Is there a C-O bond?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->N,O-DIBENZOYL2-AMINOBUTANOL<- )
p-5-7   WITH A PROBABILITY OF   91
p-12   WITH A PROBABILITY OF   75
p-8-9   WITH A PROBABILITY OF   75
p-16   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-13   WITH A PROBABILITY OF   35
p-10-11   WITH A PROBABILITY OF   -16
p-15   WITH A PROBABILITY OF   -16
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : A A B A B B).
        For carbonyl, you chose CdO.
```

```
        For aromatic ring, you chose has benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** N,O-DIBENZOYL2-AMINORUTANOL is found on Page 7.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->FENOPROFEN<- )

Looking for a C=O in FENOPROFEN:
  Found Oxygen at atom 16.  Is it a C=O?
*    Found C=O bond in FENOPROFEN at atom 15.
*   For carbonyl, Found that CdO exists. [FENOPROFEN]

*    Found Benzene (Benz) in FENOPROFEN at atom 1: ring - (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [FENOPROFEN]

Looking for a -OH in FENOPROFEN:
  Found Oxygen at atom 16.
  Found Oxygen at atom 17.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 18.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
Looking for a -OH in FENOPROFEN:
  Found Oxygen at atom 16.
  Found Oxygen at atom 17.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 18.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in FENOPROFEN:
Looking for an Amine in FENOPROFEN:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in FENOPROFEN:
Looking for a Primary Amine (NRH2) in FENOPROFEN:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in FENOPROFEN:
  Found Oxygen at atom 16.  Is it a C=O?
  Have C=O at atom 15.  Is there a C-O bond?
  Have C-O at atom 17.  Is there a -OH bond?
*    Found Carboxylic Acid (COOH) in FENOPROFEN at atom 15.
*   For carboxylic-acid, Found that COOH exists. [FENOPROFEN]

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
              (molecule name is ->FENOPROFEN<- )
p-8-9   WITH A PROBABILITY OF   91
p-13   WITH A PROBABILITY OF   75
p-5-7   WITH A PROBABILITY OF   75
p-15   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-12   WITH A PROBABILITY OF   35
p-10-11   WITH A PROBABILITY OF   -16
p-16   WITH A PROBABILITY OF   -16
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose a chiral : A A B B B A).
```



```
        For carbonyl, you chose CdO.
        For aromatic ring, you chose has benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose COOH.
***** FENOPROFEN is found on Page 8.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function grou.
       (molecule name is ->METHYL-3-BENZOYLOXYPROPANOATE<- )

Looking for a C=O in METHYL-3-BENZOYLOXYPROPANOATE:
  Found Oxygen at atom 5.  Is it a C=O?
*    Found C=O bond in METHYL-3-BENZOYLOXYPROPANOATE at atom 4.
*   For carbonyl, Found that CdO exists. [METHYL-3-BENZOYLOXYPROPANOATE]

*    Found Benzene (Benz) in METHYL-3-BENZOYLOXYPROPANOATE at atom 11: ring -
   (11 12 13 14 15 16).
*   For aromatic-ring, Found that has-benz exists. [METHYL-3-BENZOYLOXYPROPANOATE]

Looking for a -OH in METHYL-3-BENZOYLOXYPROPANOATE:
  Found Oxygen at atom 5.
  Found Oxygen at atom 6.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 8.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 10.
Looking for a -OH in METHYL-3-BENZOYLOXYPROPANOATE:
  Found Oxygen at atom 5.
  Found Oxygen at atom 6.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 8.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 10.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in METHYL-3-BENZOYLOXYPROPANOATE:
Looking for an Amine in METHYL-3-BENZOYLOXYPROPANOATE:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in METHYL-3-BENZOYLOXYPROPANOATE:
Looking for a Primary Amine (NRH2) in METHYL-3-BENZOYLOXYPROPANOATE:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in METHYL-3-BENZOYLOXYPROPANOATE:
  Found Oxygen at atom 5.  Is it a C=O?
  Have C=O at atom 4.  Is there a C-O bond?
  Have C-O at atom 6.  Is there a -OH bond?
  Found Oxygen at atom 6.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
  Have C-O at atom 8.  Is there a -OH bond?
Looking for a Carboxylic acid (XC=OOH) in METHYL-3-BENZOYLOXYPROPANOATE:
  Found Oxygen at atom 5.  Is it a C=O?
  Have C=O at atom 4.  Is there a C-O bond?
  Have C-O at atom 6.  Is there a -OH bond?
  Found Oxygen at atom 6.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
  Have C-O at atom 8.  Is there a -OH bond?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.
```

```
Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->METHYL-3-BENZOYLOXYPROPANOATE<- )
p-8-9    WITH A PROBABILITY OF    91
p-13    WITH A PROBABILITY OF    75
p-5-7    WITH A PROBABILITY OF    75
p-16    WITH A PROBABILITY OF    55
p-4    WITH A PROBABILITY OF    55
p-12    WITH A PROBABILITY OF    35
p-10-11    WITH A PROBABILITY OF    -16
p-15    WITH A PROBABILITY OF    -16
p-14    WITH A PROBABILITY OF    -68

The following is the data for (choose-a-chiral : A A B B B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** METHYL-3-BENZOYLOXYPROPANOATE is found on Page 8.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function grou.
       (molecule name is ->FLUBIPROFEN ME ESTER<- )

Looking for a C=O in FLUBIPROFEN ME ESTER:
  Found Oxygen at atom 16.  Is it a C=O?
*    Found C=O bond in FLUBIPROFEN ME ESTER at atom 15.
*   For carbonyl, Found that CdO exists. [FLUBIPROFEN ME ESTER]

*    Found Benzene (Benz) in FLUBIPROFEN ME ESTER at atom 1: ring -
   (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [FLUBIPROFEN ME ESTER]

Looking for a -OH in FLUBIPROFEN ME ESTER:
  Found Oxygen at atom 16.
  Found Oxygen at atom 17.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
Looking for a -OH in FLUBIPROFEN ME ESTER:
  Found Oxygen at atom 16.
  Found Oxygen at atom 17.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in FLUBIPROFEN ME ESTER:
Looking for an Amine in FLUBIPROFEN ME ESTER:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in FLUBIPROFEN ME ESTER:
Looking for a Primary Amine (NRH2) in FLUBIPROFEN ME ESTER:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC-OOH) in FLUBIPROFEN ME ESTER:
  Found Oxygen at atom 16.  Is it a C=O?
  Have C=O at atom 15.  Is there a C-O bond?
  Have C-O at atom 17.  Is there a -OH bond?
  Found Oxygen at atom 17.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in FLUBIPROFEN ME ESTER:
  Found Oxygen at atom 16.  Is it a C=O?
  Have C=O at atom 15.  Is there a C-O bond?
  Have C-O at atom 17.  Is there a -OH bond?
  Found Oxygen at atom 17.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->FLUBIPROFEN ME ESTER<- )
p-8-9    WITH A PROBABILITY OF    91
p-13    WITH A PROBABILITY OF    75
p-5-7    WITH A PROBABILITY OF    75
p-16    WITH A PROBABILITY OF    55
p-4    WITH A PROBABILITY OF    55
p-12    WITH A PROBABILITY OF    35
p-10-11    WITH A PROBABILITY OF    16
p-15    WITH A PROBABILITY OF    -16
p-14    WITH A PROBABILITY OF    -68

The following is the data for (choose-a-chiral : A A B B B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** FLUBIPROFEN ME ESTER is found on Page 8.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE<- )

Looking for a C=O in METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
  Found Oxygen at atom 2.  Is it a C=O?
*    Found C=O bond in METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE at atom
   1.
*   For carbonyl, Found that CdO exists. [METHYL
2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE]

*    Found Benzene (Benz) in METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE at
atom
   8: ring - (8 9 10 11 12 13).
*   For aromatic-ring, Found that has-benz exists. [METHYL
2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE]

Looking for a -OH in METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
  Found Oxygen at atom 2.
  Found Oxygen at atom 3.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 7.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 14.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
Looking for a -OH in METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
  Found Oxygen at atom 2.
  Found Oxygen at atom 3.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 7.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 14.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
```

```
*    For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
Looking for an Amine in METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
*    For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in METHYL
2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
Looking for a Primary Amine (NRH2) in METHYL
2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
*    For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in METHYL
2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
  Found Oxygen at atom 2.  Is it a C=O?
  Have C=O at atom 1.  Is there a C-O bond?
  Have C-O at atom 3.  Is there a -OH bond?
  Found Oxygen at atom 3.  Is it a C=O?
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 14.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in METHYL
2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE:
  Found Oxygen at atom 2.  Is it a C=O?
  Have C=O at atom 1.  Is there a C-O bond?
  Have C-O at atom 3.  Is there a -OH bond?
  Found Oxygen at atom 3.  Is it a C=O?
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 14.  Is it a C=O?
*    For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->METHYL
2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE<- )
p-8-9    WITH A PROBABILITY OF    91
p-13    WITH A PROBABILITY OF    75
p-5-7    WITH A PROBABILITY OF    75
p-16    WITH A PROBABILITY OF    55
p-4    WITH A PROBABILITY OF    55
p-12    WITH A PROBABILITY OF    35
p-10-11    WITH A PROBABILITY OF    -16
p-15    WITH A PROBABILITY OF    -16
p-14    WITH A PROBABILITY OF    -68

The following is the data for (choose-a-chiral : A A B B B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE is
on Page 9.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function grou.
       (molecule name is ->METHYL2-PHENYLPROPIONATE<- )

Looking for a C=O in METHYL2 PHENYLPROPIONATE:
  Found Oxygen at atom 10.  Is it a C=O?
*    Found C=O bond in METHYL2-PHENYLPROPIONATE at atom 9.
*    For carbonyl, Found that CdO exists. [METHYL2-PHENYLPROPIONATE]

*    Found Benzene (Benz) in METHYL2-PHENYLPROPIONATE at atom 1: ring -
   (1 2 3 4 5 6).
*    For aromatic-ring, Found that has-benz exists. [METHYL2-PHENYLPROPIONATE]

Looking for a -OH in METHYL2-PHENYLPROPIONATE:
  Found Oxygen at atom 10.
  Found Oxygen at atom 11.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
Looking for a -OH in METHYL2-PHENYLPROPIONATE:
  Found Oxygen at atom 10.
  Found Oxygen at atom 11.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in METHYL2-PHENYLPROPIONATE:
Looking for an Amine in METHYL2-PHENYLPROPIONATE:
*    For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in METHYL2-PHENYLPROPIONATE:
Looking for a Primary Amine (NRH2) in METHYL2-PHENYLPROPIONATE:
*    For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in METHYL2-PHENYLPROPIONATE:
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
  Have C-O at atom 11.  Is there a -OH bond?
  Found Oxygen at atom 11.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in METHYL2-PHENYLPROPIONATE:
  Found Oxygen at atom 10.  Is it a C=O?
  Have C=O at atom 9.  Is there a C-O bond?
  Have C-O at atom 11.  Is there a -OH bond?
  Found Oxygen at atom 11.  Is it a C=O?
*    For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->METHYL2-PHENYLPROPIONATE<- )
p-8-9    WITH A PROBABILITY OF    91
p-13    WITH A PROBABILITY OF    75
p-5-7    WITH A PROBABILITY OF    75
p-16    WITH A PROBABILITY OF    55
p-4    WITH A PROBABILITY OF    55
p-12    WITH A PROBABILITY OF    35
p-10-11    WITH A PROBABILITY OF    -16
p-15    WITH A PROBABILITY OF    -16
p-14    WITH A PROBABILITY OF    -68

The following is the data for (choose-a-chiral : A A B B B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** METHYL2-PHENYLPROPIONATE is found on Page 9.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function grou.
       (molecule name is ->1 METHYL 3 PHENYL PROPYL BENZOATE<- )

Looking for a C=O in 1 METHYL 3 PHENYL PROPYL BENZOATE:
  Found Oxygen at atom 10.  Is it a C=O?
  Found Oxygen at atom 12.  Is it a C=O?
```

```
*    Found C-O bond in 1 METHYL 3 PHENYL PROPYL BENZOATE at atom 11.
*   For carbonyl, Found that CdO exists. [1 METHYL 3 PHENYL PROPYL BENZOATE]

*    Found Benzene (Benz) in 1 METHYL 3 PHENYL PROPYL BENZOATE at atom 1: ring -
   (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [1 METHYL 3 PHENYL PROPYL
BENZOATE]

Looking for a -OH in 1 METHYL 3 PHENYL PROPYL BENZOATE:
  Found Oxygen at atom 10.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 12.
Looking for a -OH in 1 METHYL 3 PHENYL PROPYL BENZOATE:
  Found Oxygen at atom 10.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 12.
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in 1 METHYL 3 PHENYL PROPYL BENZOATE:
Looking for an Amine in 1 METHYL 3 PHENYL PROPYL BENZOATE:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in 1 METHYL 3 PHENYL PROPYL BENZOATE:
Looking for a Primary Amine (NRH2) in 1 METHYL 3 PHENYL PROPYL BENZOATE:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in 1 METHYL 3 PHENYL PROPYL BENZOATE:
  Found Oxygen at atom 10.  Is it a C=O?
  Found Oxygen at atom 12.  Is it a C=O?
  Have C=O at atom 11.  Is there a C-O bond?
  Have C-O at atom 10.  Is there a -OH bond?
Looking for a Carboxylic acid (XC=OOH) in 1 METHYL 3 PHENYL PROPYL BENZOATE:
  Found Oxygen at atom 10.  Is it a C=O?
  Found Oxygen at atom 12.  Is it a C=O?
  Have C=O at atom 11.  Is there a C-O bond?
  Have C-O at atom 10.  Is there a -OH bond?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->1 METHYL 3 PHENYL PROPYL BENZOATE<- )
p-8-9   WITH A PROBABILITY OF   91
p-13   WITH A PROBABILITY OF   75
p-5-7   WITH A PROBABILITY OF   75
p-16   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-12   WITH A PROBABILITY OF   35
p-10-11   WITH A PROBABILITY OF   -16
p-15   WITH A PROBABILITY OF   -16
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : A A B B B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** 1 METHYL 3 PHENYL PROPYL BENZOATE is found on Page 9.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
```



```
      (molecule name is ->GUALFENESINE<- )

Looking for a C=O in GUALFENESINE:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 13.  Is it a C=O?
  Found Oxygen at atom 14.  Is it a C=O?
Looking for a C=O in GUALFENESINE:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 13.  Is it a C=O?
  Found Oxygen at atom 14.  Is it a C=O?
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*    Found Benzene (Benz) in GUALFENESINE at atom 1: ring - (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [GUALFENESINE]

Looking for a -OH in GUALFENESINE:
  Found Oxygen at atom 7.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 8.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 13.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in GUALFENESINE at atom 14.
*   For hydroxyl, Found that OsH exists. [GUALFENESINE]

Looking for an Amine in GUALFENESINE:
Looking for an Amine in GUALFENESINE:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in GUALFENESINE:
Looking for a Primary Amine (NRH2) in GUALFENESINE:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in GUALFENESINE:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 13.  Is it a C=O?
  Found Oxygen at atom 14.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in GUALFENESINE:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 8.  Is it a C=O?
  Found Oxygen at atom 13.  Is it a C=O?
  Found Oxygen at atom 14.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->GUALFENESINE<- )
p-10-11   WITH A PROBABILITY OF   83
p-13   WITH A PROBABILITY OF   75
p-4   WITH A PROBABILITY OF   55
p-16   WITH A PROBABILITY OF   55
p-8-9   WITH A PROBABILITY OF   35
p-12   WITH A PROBABILITY OF   35
p-15   WITH A PROBABILITY OF   -16
p-5-7   WITH A PROBABILITY OF   -53
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : B A A B B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
```

```
        For hydroxyl, you chose OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** GUAIFENESINE is found on Page 10.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE<- )

Looking for a C=O in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE:
  Found Oxygen at atom 16.  Is it a C=O?
Looking for a C=O in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE:
  Found Oxygen at atom 16.  Is it a C=O?
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*    Found Benzene (Benz) in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE at atom
   1: ring - (1 2 5 9 7 3).
*   For aromatic-ring, Found that has-benz exists. [9 (2-TRIFLORO 1-HYDROXY) ETHYL
ANTHRACENE]

Looking for a -OH in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE:
  Found Oxygen at atom 16.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE at atom 16.
*   For hydroxyl, Found that OsH exists. [9 (2-TRIFLORO 1-HYDROXY) ETHYL
ANTHRACENE]

Looking for an Amine in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE:
Looking for an Amine in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE:
Looking for a Primary Amine (NRH2) in 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in 9 (2-TRIFLORO 1-HYDROXY) ETHYL
ANTHRACENE:
  Found Oxygen at atom 16.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in 9 (2-TRIFLORO 1-HYDROXY) ETHYL
ANTHRACENE:
  Found Oxygen at atom 16.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE<- )
p-10-11   WITH A PROBABILITY OF   83
p-13   WITH A PROBABILITY OF   75
p-4   WITH A PROBABILITY OF   55
p-16   WITH A PROBABILITY OF   55
p-8-9   WITH A PROBABILITY OF   35
p-12   WITH A PROBABILITY OF   35
p-15   WITH A PROBABILITY OF   -16
p-5-7   WITH A PROBABILITY OF   -53
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : B A A B B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.

***** 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE is
on Page 9.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->1-PHENYL-1,2-ETHANEDIOL<- )

Looking for a C=O in 1-PHENYL-1,2-ETHANEDIOL:
  Found Oxygen at atom 9.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
Looking for a C=O in 1-PHENYL-1,2-ETHANEDIOL:
  Found Oxygen at atom 9.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*    Found Benzene (Benz) in 1-PHENYL-1,2-ETHANEDIOL at atom 1: ring -
   (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [1-PHENYL-1,2-ETHANEDIOL]

Looking for a -OH in 1-PHENYL-1,2-ETHANEDIOL:
  Found Oxygen at atom 9.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in 1-PHENYL-1,2-ETHANEDIOL at atom 9.
*   For hydroxyl, Found that OsH exists. [1-PHENYL-1,2-ETHANEDIOL]

Looking for an Amine in 1-PHENYL-1,2-ETHANEDIOL:
Looking for an Amine in 1-PHENYL-1,2-ETHANEDIOL:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in 1-PHENYL-1,2-ETHANEDIOL:
Looking for a Primary Amine (NRH2) in 1-PHENYL-1,2-ETHANEDIOL:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in 1-PHENYL-1,2-ETHANEDIOL:
  Found Oxygen at atom 9.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in 1-PHENYL-1,2-ETHANEDIOL:
  Found Oxygen at atom 9.  Is it a C=O?
  Found Oxygen at atom 10.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->1-PHENYL-1,2-ETHANEDIOL<- )
p-10-11   WITH A PROBABILITY OF   83
p-13   WITH A PROBABILITY OF   75
p-4   WITH A PROBABILITY OF   55
p-16   WITH A PROBABILITY OF   55
p-8-9   WITH A PROBABILITY OF   35
p-12   WITH A PROBABILITY OF   35
p-15   WITH A PROBABILITY OF   -16
p-5-7   WITH A PROBABILITY OF   -53
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : B A A B B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** 1 PHENYL 1,2-ETHANEDIOL is found on Page 10.
```

```
We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function grou.
       (molecule name is ->PERISOXAL<- )

Looking for a C=O in PERISOXAL:
  Found Oxygen at atom 8.  Is it a C-O?
  Found Oxygen at atom 14.  Is it a C=O?
Looking for a C=O in PERISOXAL:
  Found Oxygen at atom 8.  Is it a C-O?
  Found Oxygen at atom 14.  Is it a C=O?
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*    Found Benzene (Benz) in PERISOXAL at atom 1: ring - (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [PERISOXAL]

Looking for a -OH in PERISOXAL:
  Found Oxygen at atom 8.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 14.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in PERISOXAL at atom 14.
*   For hydroxyl, Found that OsH exists. [PERISOXAL]

Looking for an Amine in PERISOXAL:
*    Found Amine (-NR2) in PERISOXAL at atom 18.
*   For amine, Found that N3 exists. [PERISOXAL]

Looking for a Primary Amine (NRH2) in PERISOXAL:
Looking for a Primary Amine (NRH2) in PERISOXAL:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in PERISOXAL:
  Found Oxygen at atom 8.  Is it a C-O?
  Found Oxygen at atom 14.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in PERISOXAL:
  Found Oxygen at atom 8.  Is it a C-O?
  Found Oxygen at atom 14.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->PERISOXAL<- )
p-10-11   WITH A PROBABILITY OF   83
p-12   WITH A PROBABILITY OF   75
p-4   WITH A PROBABILITY OF   55
p-16   WITH A PROBABILITY OF   55
p-5-7   WITH A PROBABILITY OF   35
p-13   WITH A PROBABILITY OF   35
p-15   WITH A PROBABILITY OF   -16
p-8-9   WITH A PROBABILITY OF   -53
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : B A A A B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** PERISOXAL is found on Page 11.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->PINDOLOL<- )

Looking for a C=O in PINDOLOL:
  Found Oxygen at atom 9.  Is it a C-O?
  Found Oxygen at atom 13.  Is it a C=O?
Looking for a C=O in PINDOLOL:
  Found Oxygen at atom 9.  Is it a C=O?
  Found Oxygen at atom 13.  Is it a C-O?
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*    Found Benzene (Benz) in PINDOLOL at atom 1: ring - (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [PINDOLOL]

Looking for a -OH in PINDOLOL:
  Found Oxygen at atom 9.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 13.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in PINDOLOL at atom 13.
*   For hydroxyl, Found that OsH exists. [PINDOLOL]

Looking for an Amine in PINDOLOL:
*    Found Amine (-NR2) in PINDOLOL at atom 8.
*   For amine, Found that N3 exists. [PINDOLOL]

Looking for a Primary Amine (NRH2) in PINDOLOL:
Looking for a Primary Amine (NRH2) in PINDOLOL:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in PINDOLOL:
  Found Oxygen at atom 9.  Is it a C-O?
  Found Oxygen at atom 13.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in PINDOLOL:
  Found Oxygen at atom 9.  Is it a C=O?
  Found Oxygen at atom 13.  Is it a C-O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->PINDOLOL<- )
p-10-11   WITH A PROBABILITY OF   83
p-12   WITH A PROBABILITY OF   75
p-4   WITH A PROBABILITY OF   55
p-16   WITH A PROBABILITY OF   55
p-5-7   WITH A PROBABILITY OF   35
p-13   WITH A PROBABILITY OF   35
p-15   WITH A PROBABILITY OF   -16
p-8-9   WITH A PROBABILITY OF   -53
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : B A A A B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** PINDOLOL is found on Page 11.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->METOPROLOL<- )
```

```
Looking for a C=O in METOPROLOL:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 11.  Is it a C=O?
  Found Oxygen at atom 18.  Is it a C=O?
Looking for a C-O in METOPROLOL:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 11.  Is it a C=O?
  Found Oxygen at atom 18.  Is it a C=O?
*    For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*     Found Benzene (Benz) in METOPROLOL at atom 1: ring - (1 2 3 4 5 6).
*    For aromatic-ring, Found that has-benz exists. [METOPROLOL]

Looking for a -OH in METOPROLOL:
  Found Oxygen at atom 7.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 11.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*     Found  OH bond in METOPROLOL at atom 11.
*    For hydroxyl, Found that OsH exists. [METOPROLOL]

Looking for an Amine in METOPROLOL:
*     Found Amine ( NR2) in METOPROLOL at atom 12.
*    For amine, Found that N3 exists. [METOPROLOL]

Looking for a Primary Amine (NRH2) in METOPROLOL:
Looking for a Primary Amine (NRH2) in METOPROLOL:
*    For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in METOPROLOL:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 11.  Is it a C=O?
  Found Oxygen at atom 18.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in METOPROLOL:
  Found Oxygen at atom 7.  Is it a C=O?
  Found Oxygen at atom 11.  Is it a C=O?
  Found Oxygen at atom 18.  Is it a C=O?
*    For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->METOPROLOL<- )
p-10-11    WITH A PROBABILITY OF    83
p-12    WITH A PROBABILITY OF    75
p-4    WITH A PROBABILITY OF    55
p-16    WITH A PROBABILITY OF    55
p-5-7    WITH A PROBABILITY OF    35
p-13    WITH A PROBABILITY OF    35
p-15    WITH A PROBABILITY OF    -16
p-8-9    WITH A PROBABILITY OF    -53
p-14    WITH A PROBABILITY OF    -68

The following is the data for (choose-a-chiral : B A A A B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** METOPROLOL is found on Page 11.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
```



```
               (molecule name is ->CARBINOXAMINE<- )

Looking for a C=O in CARBINOXAMINE:
  Found Oxygen at atom 9.  Is it a C=O?
Looking for a C=O in CARBINOXAMINE:
  Found Oxygen at atom 9.  Is it a C=O?
*    For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*     Found Benzene (Benz) in CARBINOXAMINE at atom 1: ring - (1 2 3 4 5 6).
*    For aromatic-ring, Found that has-benz exists. [CARBINOXAMINE]

Looking for a -OH in CARBINOXAMINE:
  Found Oxygen at atom 9.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
Looking for a -OH in CARBINOXAMINE:
  Found Oxygen at atom 9.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in CARBINOXAMINE:
*     Found Amine (-NR2) in CARBINOXAMINE at atom 18.
*    For amine, Found that N3 exists. [CARBINOXAMINE]

Looking for a Primary Amine (NRH2) in CARBINOXAMINE:
Looking for a Primary Amine (NRH2) in CARBINOXAMINE:
*    For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in CARBINOXAMINE:
  Found Oxygen at atom 9.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in CARBINOXAMINE:
  Found Oxygen at atom 9.  Is it a C=O?
*    For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->CARBINOXAMINE<- )
p-12    WITH A PROBABILITY OF    91
p-13    WITH A PROBABILITY OF    75
p-5-7    WITH A PROBABILITY OF    75
p-16    WITH A PROBABILITY OF    55
p-10-11    WITH A PROBABILITY OF    55
p-8-9    WITH A PROBABILITY OF    35
p-4    WITH A PROBABILITY OF    -16
p-15    WITH A PROBABILITY OF    -16
p-14    WITH A PROBABILITY OF    -68

The following is the data for (choose-a-chiral : B A B A B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** CARBINOXAMINE is found on Page 12.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->1-PHENYLETHYLAMINE<- )

Looking for a C=O in 1-PHENYLETHYLAMINE:
Looking for a C=O in 1-PHENYLETHYLAMINE:
*    For carbonyl, Found CdO dosn't exist, so it's no-CdO.
```

```
*    Found Benzene (Benz) in 1-PHENYLETHYLAMINE at atom 4: ring - (4 5 6 7 8 9).
*   For aromatic-ring, Found that has-benz exists. [1-PHENYLETHYLAMINE]

Looking for a -OH in 1-PHENYLETHYLAMINE:
Looking for a -OH in 1-PHENYLETHYLAMINE:
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in 1-PHENYLETHYLAMINE:
*    Found Amine (-NR2) in 1-PHENYLETHYLAMINE at atom 3.
*   For amine, Found that N3 exists. [1-PHENYLETHYLAMINE]

Looking for a Primary Amine (NRH2) in 1-PHENYLETHYLAMINE:
*    Found Primary Amine (-NH2) in 1-PHENYLETHYLAMINE at atom 3.
*   For primary-amine, Found that NP exists. [1-PHENYLETHYLAMINE]

Looking for a Carboxylic acid (XC-OOH) in 1-PHENYLETHYLAMINE:
Looking for a Carboxylic acid (XC-OOH) in 1-PHENYLETHYLAMINE:
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->1-PHENYLETHYLAMINE<- )
p-12   WITH A PROBABILITY OF   91
p-13   WITH A PROBABILITY OF   75
p-5-7   WITH A PROBABILITY OF   75
p-10-11   WITH A PROBABILITY OF   55
p-8-9   WITH A PROBABILITY OF   35
p-14   WITH A PROBABILITY OF   20
p-16   WITH A PROBABILITY OF   6
p-4   WITH A PROBABILITY OF   -16
p-15   WITH A PROBABILITY OF   -82

The following is the data for (choose-a-chiral : B A B A A B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose no-COOH.
***** 1-PHENYLETHYLAMINE is found on Page 12.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->1-BROMO ETHYLBENZENE<- )

Looking for a C=O in 1-BROMO ETHYLBENZENE:
Looking for a C=O in 1-BROMO ETHYLBENZENE:
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*    Found Benzene (Benz) in 1-BROMO ETHYLBENZENE at atom 1: ring -
   (1 2 3 4 5 6).
*   For aromatic-ring, Found that has-benz exists. [1-BROMO ETHYLBENZENE]

Looking for a -OH in 1-BROMO ETHYLBENZENE:
Looking for a -OH in 1-BROMO ETHYLBENZENE:
*   For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in 1-BROMO ETHYLBENZENE:
Looking for an Amine in 1-BROMO ETHYLBENZENE:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in 1-BROMO ETHYLBENZENE:
Looking for a Primary Amine (NRH2) in 1-BROMO ETHYLBENZENE:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC-OOH) in 1-BROMO ETHYLBENZENE:
Looking for a Carboxylic acid (XC-OOH) in 1-BROMO ETHYLBENZENE:
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->1-BROMO ETHYLBENZENE<- )
p-13   WITH A PROBABILITY OF   91
p-12   WITH A PROBABILITY OF   75
p-8-9   WITH A PROBABILITY OF   75
p-16   WITH A PROBABILITY OF   55
p-10-11   WITH A PROBABILITY OF   55
p-5-7   WITH A PROBABILITY OF   35
p-4   WITH A PROBABILITY OF   -16
p-15   WITH A PROBABILITY OF   -16
p-14   WITH A PROBABILITY OF   -68

The following is the data for (choose-a-chiral : B A B B B B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose has-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** 1-BROMO ETHYLBENZENE is found on Page 13.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->2-AMINO-1-HEXANOL<- )

Looking for a C=O in 2-AMINO-1-HEXANOL:
  Found Oxygen at atom 3.  Is it a C=O?
Looking for a C=O in 2-AMINO-1-HEXANOL:
  Found Oxygen at atom 3.  Is it a C=O?
*   For carbonyl, Found CdO dosn't exist, so it's no-CdO.

*   For aromatic-ring, Found has-benz dosn't exist, so it's no-benz.

Looking for a -OH in 2-AMINO-1-HEXANOL:
  Found Oxygen at atom 3.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in 2-AMINO-1-HEXANOL at atom 3.
*   For hydroxyl, Found that OsH exists. [2-AMINO-1-HEXANOL]

Looking for an Amine in 2-AMINO-1-HEXANOL:
*    Found Amine (-NR2) in 2-AMINO-1-HEXANOL at atom 4.
*   For amine, Found that N3 exists. [2-AMINO-1-HEXANOL]

Looking for a Primary Amine (NRH2) in 2-AMINO-1-HEXANOL:
*    Found Primary Amine (-NH2) in 2-AMINO-1-HEXANOL at atom 4.
*   For primary-amine, Found that NP exists. [2-AMINO-1-HEXANOL]

Looking for a Carboxylic acid (XC=OOH) in 2-AMINO-1-HEXANOL:
  Found Oxygen at atom 3.  Is it a C=O?
Looking for a Carboxylic acid (XC=OOH) in 2-AMINO-1-HEXANOL:
  Found Oxygen at atom 3.  Is it a C=O?
*   For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
```

```
                (molecule name is ->2-AMINO-1-HEXANOL<- )
p-14   WITH A PROBABILITY OF   69
p-16   WITH A PROBABILITY OF   55
p-10-11   WITH A PROBABILITY OF   55
p-12   WITH A PROBABILITY OF   35
p-4   WITH A PROBABILITY OF   6
p-5-7   WITH A PROBABILITY OF   2
p-15   WITH A PROBABILITY OF   -16
p-13   WITH A PROBABILITY OF   -53
p-8-9   WITH A PROBABILITY OF   -90

The following is the data for (choose-a-chiral : B B A A A B).
        For carbonyl, you chose no-CdO.
        For aromatic-ring, you chose no-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose no-COOH.
***** 2-AMINO-1-HEXANOL is found on Page 14.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->SERINE<- )

Looking for a C=O in SERINE:
  Found Oxygen at atom 5.  Is it a C=O?
*    Found C=O bond in SERINE at atom 4.
*   For carbonyl, Found that CdO exists. [SERINE]

*   For aromatic-ring, Found has-benz dosn't exist, so it's no-benz.

Looking for a -OH in SERINE:
  Found Oxygen at atom 5.
  Found Oxygen at atom 6.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 7.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in SERINE at atom 7.
*   For hydroxyl, Found that OsH exists. [SERINE]

Looking for an Amine in SERINE:
*    Found Amine (-NR2) in SERINE at atom 1.
*   For amine, Found that N3 exists. [SERINE]

Looking for a Primary Amine (NRH2) in SERINE:
*    Found Primary Amine (-NH2) in SERINE at atom 1.
*   For primary-amine, Found that NP exists. [SERINE]

Looking for a Carboxylic acid (XC=OOH) in SERINE:
  Found Oxygen at atom 5.  Is it a C=O?
  Have C=O at atom 4.  Is there a C-O bond?
  Have C-O at atom 6.  Is there a -OH bond?
*    Found Carboxylic Acid (COOH) in SERINE at atom 4.
*   For carboxylic-acid, Found that COOH exists. [SERINE]

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->SERINE<- )
p-14   WITH A PROBABILITY OF   69
p-15   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-5-7   WITH A PROBABILITY OF   35
p-10-11   WITH A PROBABILITY OF   6
p-12   WITH A PROBABILITY OF   -2
p-16   WITH A PROBABILITY OF   -16
p-8-9   WITH A PROBABILITY OF   -53
p-13   WITH A PROBABILITY OF   -90

The following is the data for (choose-a-chiral : A B A A A A).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose no-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose COOH.
***** SERINE is found on Page 14.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->THREONINE<- )

Looking for a C=O in THREONINE:
  Found Oxygen at atom 5.  Is it a C=O?
*    Found C=O bond in THREONINE at atom 4.
*   For carbonyl, Found that CdO exists. [THREONINE]

*   For aromatic-ring, Found has-benz dosn't exist, so it's no-benz.

Looking for a -OH in THREONINE:
  Found Oxygen at atom 5.
  Found Oxygen at atom 6.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 8.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in THREONINE at atom 8.
*   For hydroxyl, Found that OsH exists. [THREONINE]

Looking for an Amine in THREONINE:
*    Found Amine (-NR2) in THREONINE at atom 1.
*   For amine, Found that N3 exists. [THREONINE]

Looking for a Primary Amine (NRH2) in THREONINE:
*    Found Primary Amine ( NH2) in THREONINE at atom 1.
*   For primary-amine, Found that NP exists. [THREONINE]

Looking for a Carboxylic acid (XC=OOH) in THREONINE:
  Found Oxygen at atom 5.  Is it a C=O?
  Have C=O at atom 4.  Is there a C-O bond?
  Have C-O at atom 6.  Is there a -OH bond?
*    Found Carboxylic Acid (COOH) in THREONINE at atom 4.
*   For carboxylic-acid, Found that COOH exists. [THREONINE]

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->THREONINE<- )
p-14   WITH A PROBABILITY OF   69
p-15   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-5-7   WITH A PROBABILITY OF   35
p-10-11   WITH A PROBABILITY OF   6
p-12   WITH A PROBABILITY OF   -2
p-16   WITH A PROBABILITY OF   -16
p-8-9   WITH A PROBABILITY OF   -53
p-13   WITH A PROBABILITY OF   -90

The following is the data for (choose-a-chiral : A B A A A A).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose no-benz.
```

```
        For hydroxyl, you chose OsH.
        For amine, you chose N3.
        For primary-amine, you chose NP.
        For carboxylic-acid, you chose COOH.
***** THREONINE is found on Page 14.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->LACTIC ACID<- )

Looking for a C=O in LACTIC ACID:
  Found Oxygen at atom 3.  Is it a C=O?
  Found Oxygen at atom 5.  Is it a C=O?
*    Found C=O bond in LACTIC ACID at atom 4.
*   For carbonyl, Found that CdO exists. [LACTIC ACID]

*   For aromatic-ring, Found has-benz dosn't exist, so it's no-benz.

Looking for a -OH in LACTIC ACID:
  Found Oxygen at atom 3.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in LACTIC ACID at atom 3.
*   For hydroxyl, Found that OsH exists. [LACTIC ACID]

Looking for an Amine in LACTIC ACID:
Looking for an Amine in LACTIC ACID:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in LACTIC ACID:
Looking for a Primary Amine (NRH2) in LACTIC ACID:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in LACTIC ACID:
  Found Oxygen at atom 3.  Is it a C=O?
  Found Oxygen at atom 5.  Is it a C=O?
  Have C=O at atom 4.  Is there a C-O bond?
  Have C-O at atom 6.  Is there a -OH bond?
*    Found Carboxylic Acid (COOH) in LACTIC ACID at atom 4.
*   For carboxylic-acid, Found that COOH exists. [LACTIC ACID]

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->LACTIC ACID<- )
p-15   WITH A PROBABILITY OF   83
p-16   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-8-9   WITH A PROBABILITY OF   35
p-14   WITH A PROBABILITY OF   20
p-10-11   WITH A PROBABILITY OF   6
p-13   WITH A PROBABILITY OF   -2
p-5-7   WITH A PROBABILITY OF   -53
p-12   WITH A PROBABILITY OF   -90

The following is the data for (choose-a-chiral : A B A B B A).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose no-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose COOH.
***** LACTIC ACID is found on Page 15.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->GLYCERIC ACID<-  )

Looking for a C=O in GLYCERIC ACID:
  Found Oxygen at atom 3.  Is it a C=O?
  Found Oxygen at atom 4.  Is it a C=O?
  Found Oxygen at atom 6.  Is it a C=O?
*    Found C=O bond in GLYCERIC ACID at atom 5.
*   For carbonyl, Found that CdO exists. [GLYCERIC ACID]

*   For aromatic-ring, Found has-benz dosn't exist, so it's no-benz.

Looking for a -OH in GLYCERIC ACID:
  Found Oxygen at atom 3.
  Single bond w/ 'O' found.  Look for bond w/another atom.
*    Found -OH bond in GLYCERIC ACID at atom 3.
*   For hydroxyl, Found that OsH exists. [GLYCERIC ACID]

Looking for an Amine in GLYCERIC ACID:
Looking for an Amine in GLYCERIC ACID:
*   For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in GLYCERIC ACID:
Looking for a Primary Amine (NRH2) in GLYCERIC ACID:
*   For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in GLYCERIC ACID:
  Found Oxygen at atom 3.  Is it a C=O?
  Found Oxygen at atom 4.  Is it a C=O?
  Found Oxygen at atom 6.  Is it a C=O?
  Have C=O at atom 5.  Is there a C-O bond?
  Have C-O at atom 7.  Is there a -OH bond?
*    Found Carboxylic Acid (COOH) in GLYCERIC ACID at atom 5.
*   For carboxylic-acid, Found that COOH exists. [GLYCERIC ACID]

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
               (molecule name is ->GLYCERIC ACID<- )
p-15   WITH A PROBABILITY OF   83
p-16   WITH A PROBABILITY OF   55
p-4   WITH A PROBABILITY OF   55
p-8-9   WITH A PROBABILITY OF   35
p-14   WITH A PROBABILITY OF   20
p-10-11   WITH A PROBABILITY OF   6
p-13   WITH A PROBABILITY OF   -2
p-5-7   WITH A PROBABILITY OF   -53
p-12   WITH A PROBABILITY OF   -90

The following is the data for (choose-a-chiral : A B A B B A).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose no-benz.
        For hydroxyl, you chose OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose COOH.
***** GLYCERIC ACID is found on Page 15.


We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on function-grou.
       (molecule name is ->1,3-DIACETOXY-BUTANE<- )

Looking for a C=O in 1,3-DIACETOXY-BUTANE:
  Found Oxygen at atom 5.  Is it a C=O?
  Found Oxygen at atom 7.  Is it a C=O?
```

```
*     Found C=O bond in 1.3-DIACETOXY-BUTANE at atom 6.
*    For carbonyl, Found that CdO exists. [1.3-DIACETOXY-BUTANE]

*    For aromatic-ring, Found has-benz dosn't exist, so it's no benz.

Looking for a -OH in 1.3-DIACETOXY-BUTANE:
  Found Oxygen at atom 5.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 7.
  Found Oxygen at atom 9.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 11.
Looking for a -OH in 1.3-DIACETOXY-BUTANE:
  Found Oxygen at atom 5.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 7.
  Found Oxygen at atom 9.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Single bond w/ 'O' found.  Look for bond w/another atom.
  Found Oxygen at atom 11.
*    For hydroxyl, Found OsH dosn't exist, so it's no-OsH.

Looking for an Amine in 1.3-DIACETOXY-BUTANE:
Looking for an Amine in 1.3-DIACETOXY-BUTANE:
*    For amine, Found N3 dosn't exist, so it's no-N3.

Looking for a Primary Amine (NRH2) in 1.3-DIACETOXY-BUTANE:
Looking for a Primary Amine (NRH2) in 1.3-DIACETOXY-BUTANE:
*    For primary-amine, Found NP dosn't exist, so it's no-NP.

Looking for a Carboxylic acid (XC=OOH) in 1.3-DIACETOXY-BUTANE:
  Found Oxygen at atom 5.  Is it a C=O?
  Found Oxygen at atom 7.  Is it a C=O?
  Have C=O at atom 6.  Is there a C-O bond?
  Have C-O at atom 5.  Is there a -OH bond?
  Found Oxygen at atom 9.  Is it a C=O?
  Found Oxygen at atom 11.  Is it a C=O?
  Have C=O at atom 10.  Is there a C-O bond?
  Have C-O at atom 9.  Is there a -OH bond?
Looking for a Carboxylic acid (XC=OOH) in 1.3-DIACETOXY-BUTANE:
  Found Oxygen at atom 5.  Is it a C=O?
  Found Oxygen at atom 7.  Is it a C=O?
  Have C=O at atom 6.  Is there a C-O bond?
  Have C-O at atom 5.  Is there a -OH bond?
  Found Oxygen at atom 9.  Is it a C=O?
  Found Oxygen at atom 11.  Is it a C=O?
  Have C=O at atom 10.  Is there a C-O bond?
  Have C-O at atom 9.  Is there a -OH bond?
*    For carboxylic-acid, Found COOH dosn't exist, so it's no-COOH.

Finished asking questions!!

End of Rules!

We are currently looking at choose-a-chiral, trying to decide on
     which a-column to choose based on functional-grou.
                (molecule name is ->1.3-DIACETOXY-BUTANE<- )
p-16    WITH A PROBABILITY OF   83
p-8-9    WITH A PROBABILITY OF   75
p-15    WITH A PROBABILITY OF   55
p-13    WITH A PROBABILITY OF   48
p-5-7    WITH A PROBABILITY OF   35
p-14    WITH A PROBABILITY OF   20
p-4    WITH A PROBABILITY OF   -16
p-12    WITH A PROBABILITY OF   -35
p-10-11    WITH A PROBABILITY OF   -82

The following is the data for (choose-a-chiral : A B B B B B).
        For carbonyl, you chose CdO.
        For aromatic-ring, you chose no-benz.
        For hydroxyl, you chose no-OsH.
        For amine, you chose no-N3.
        For primary-amine, you chose no-NP.
        For carboxylic-acid, you chose no-COOH.
***** 1.3-DIACETOXY-BUTANE is found on Page 16.


   YES

  STAUFFERST    job terminated at 14-AUG-1993 13:52:16.19

  Accounting Information:
  Buffered I/O count:              157          Peak working set size:     2044
  Direct I/O count:                358          Peak page file size:      10304
  Page faults:                   10223          Mounted volumes:              0
  Charged CPU time:      0 00:08:22.08          Elapsed time:      0 00:08:54.16
```

APPENDIX I: CHIRULE Testing Results

Appendix I contains the log of the results summarized in Table XIII. These results are discussed in Section VII.A.1.

```
        Started:                  Finished:            Total Time:
 14-AUG-1993 11:47:41    14-AUG-1993 16:22:49          6:35:08

hcc -a500K -h2M -n test-cr.hc

************************************************************
*** Listing all molecules in the MOL Knowledge base .....
     Found Molecule (1) MANDELIC ACID
     Found Molecule (2) MANDELAMIDE
     Found Molecule (3) METHYL MANDELATE
     Found Molecule (4) PHENYLGLYCINE
     Found Molecule (5) D-PHENYLALANINE
     Found Molecule (6) 1-PHENYLETHANOL
     Found Molecule (7) 2-PHENYLPROPIONIC ACID
     Found Molecule (8) METHYL2-PHENYLPROPIONATE
     Found Molecule (9) 2-AMINO-2-PHENYLETHANOL
     Found Molecule (10) 2-AMINO-1-PHENYLETHANOL
     Found Molecule (11) 1-PHENYLETHYLAMINE
     Found Molecule (12) 1-PHENYL-PROPAN-1-OL
     Found Molecule (13) ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL
     Found Molecule (14) 2-ACETOXY-3-PHENYLPROPANE
     Found Molecule (15) 2-AMINO-3-METHYLBUTANE
     Found Molecule (16) 2-AMINO-1-HEXANOL
     Found Molecule (17) 2-AMINO-1-PENTANOL
     Found Molecule (18) DL-TYROSINE
     Found Molecule (19) L-3,4-DIHYDROXYPHENYLALANINE
     Found Molecule (20) D-(+)-TRYPTOPHAN
     Found Molecule (21) L-GLUTAMINE
     Found Molecule (22) 2 BUTYL PHENYL CARBAMATE
     Found Molecule (23) 2-PHENYL-PROPAN-1-OL
     Found Molecule (24) 1-PHENYL-1,2-ETHANEDIOL
     Found Molecule (25) 1 PHENYL 1,3 PROPANEDIOL
     Found Molecule (26) N,O DIBENZOYL2-AMINOBUTANOL
     Found Molecule (27) ALPHA METHYL PHENETHYL BENZOATE
     Found Molecule (28) 1,3-DIACETOXY-BUTANE
     Found Molecule (29) PRACTOLOL
     Found Molecule (30) BENZOIN
     Found Molecule (31) CARTEOLOL
     Found Molecule (32) CYCLOPENTOLATE
     Found Molecule (33) HISTIDINE
     Found Molecule (34) METHYL-3-BENZOYLOXYPROPANOATE
     Found Molecule (35) GUAIFENESINE
     Found Molecule (36) 1-(1-HYDROXYETHYL)NAPHTHALENE
     Found Molecule (37) 2-(1-HYDROXYETHYL)NAPHTHALENE
     Found Molecule (38) 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
     Found Molecule (39) CARBINOXAMINE
     Found Molecule (40) 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
     Found Molecule (41) SULCONAZOL
     Found Molecule (42) BENZYL PHENYL CARBINOL
     Found Molecule (43) PROPRANOLOL
     Found Molecule (44) PINDOLOL
     Found Molecule (45) METOPROLOL
     Found Molecule (46) ALPRENOLOL
     Found Molecule (47) PERISOXAL
     Found Molecule (48) L-METHIONINE
     Found Molecule (49) LYSINE
     Found Molecule (50) CITRULLINE
     Found Molecule (51) DL-VALINE
     Found Molecule (52) L-ISOLEUCINE
     Found Molecule (53) L-ORNITHINE
     Found Molecule (54) ARGININE
     Found Molecule (55) L-GLUTAMIC ACID
     Found Molecule (56) ASPARAGINE
     Found Molecule (57) ASPARTIC ACID
     Found Molecule (58) DL-LEUCINE
     Found Molecule (59) ACETYLPHENETHOLIDE
     Found Molecule (60) KETOPROFEN
     Found Molecule (61) FENOPROFEN
     Found Molecule (62) LACTIC ACID
     Found Molecule (63) GLYCERIC ACID
     Found Molecule (64) ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE
     Found Molecule (65) WARFARINE
     Found Molecule (66) N-AMIDE-L-PHENYLALANINE
     Found Molecule (67) N-ACETYL TRYPTOPHAN
     Found Molecule (68) N ACTEYL VALINE
     Found Molecule (69) TRIADIMEFON
     Found Molecule (70) TIAPROFENIC ACID ME ESTER
     Found Molecule (71) 1-BROMO ETHYLBENZENE
     Found Molecule (72) N-ACETYL-L-PHENYLALANINE
     Found Molecule (73) N-(PHENYLETHYL)ACETAMIDE
     Found Molecule (74) 1,3 BUTANEDIOL DIBENZOATE
     Found Molecule (75) 1,4 PENTYL DIBENZOATE
     Found Molecule (76) 1,5 HEXYL DIBENZOATE
     Found Molecule (77) 3 PHENYL BUTYL BENZOATE
     Found Molecule (78) 1 METHYL 3 PHENYL PROPYL BENZOATE
     Found Molecule (79) METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE
     Found Molecule (80) METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
     Found Molecule (81) N-CBZ-ALANINE
     Found Molecule (82) N-CBZ-NOR-LEUCINE
     Found Molecule (83) N-CBZ-VALINE
     Found Molecule (84) N-CBZ-ASPARAGINE
     Found Molecule (85) N-CBZ-PHENYLALANINE
     Found Molecule (86) 2-AMINO-3-PHENYLPROPIONAMIDE
     Found Molecule (87) 2PHENOXYPROPIONIC ACID
     Found Molecule (88) ETHYL 2,3 DIPHENYL PROPANOATE
     Found Molecule (89) 2 ETHYL PHENYL BENZOATE
     Found Molecule (90) PENTYNYL 3 BENZOATE
     Found Molecule (91) BUTYNYL 3 BENZOATE
     Found Molecule (92) METHYL 2 PHENOXYPROPIONATE
     Found Molecule (93) THREONINE
     Found Molecule (94) SERINE
     Found Molecule (95) ALANINE
     Found Molecule (96) 2 AMINO BUTANE
=== Found a total of 96 molecules.
============================================================

   YES

************************************************************
*** Testing all columns for predicate list_mols .....

************************************************************
*** Looking for molecules with column Crownpak CR ....
     Found Molecule (1) PHENYLGLYCINE
     Found Molecule (2) D-PHENYLALANINE
     Found Molecule (3) 2-AMINO-2-PHENYLETHANOL
     Found Molecule (4) 2-AMINO-1-PHENYLETHANOL
     Found Molecule (5) 1-PHENYLETHYLAMINE
     Found Molecule (6) ERYTHRO-2-AMINO-1,2 DIPHENYLETHANOL
     Found Molecule (7) 2-AMINO-3-METHYLBUTANE
     Found Molecule (8) 2-AMINO-1-HEXANOL
     Found Molecule (9) 2 AMINO-1-PENTANOL
     Found Molecule (10) DL TYROSINE
     Found Molecule (11) L-3,4-DIHYDROXYPHENYLALANINE
     Found Molecule (12) D-(+)-TRYPTOPHAN
     Found Molecule (13) L-GLUTAMINE
     Found Molecule (14) HISTIDINE
     Found Molecule (15) 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
     Found Molecule (16) L-METHIONINE
     Found Molecule (17) LYSINE
     Found Molecule (18) CITRULLINE
     Found Molecule (19) DL-VALINE
     Found Molecule (20) L-ISOLEUCINE
     Found Molecule (21) L-ORNITHINE
     Found Molecule (22) ARGININE
     Found Molecule (23) L GLUTAMIC ACID
     Found Molecule (24) ASPARAGINE
     Found Molecule (25) ASPARTIC ACID
```

```
     Found Molcule (26) THREONINE
     Found Molcule (27) SERINE
     Found Molcule (28) ALANINE
     Found Molcule (29) 2 AMINO BUTANE
=== Found a total of 29 molecules for column Crownpak CR.
==========================================================


************************************************************
*** Looking for molecules with column Chiralcel OB ....
     Found Molcule (1) 1-PHENYLETHANOL
     Found Molcule (2) 1-PHENYL-PROPAN-1-OL
     Found Molcule (3) 2-ACETOXY-3-PHENYLPROPANE
     Found Molcule (4) 2-PHENYL-PROPAN-1-OL
     Found Molcule (5) 1-PHENYL 1,2-ETHANEDIOL
     Found Molcule (6) 1 PHENYL 1,3-PROPANEDIOL
     Found Molcule (7) N,O-DIBENZOYL2-AMINOBUTANOL
     Found Molcule (8) ALPHA METHYL PHENETHYL BENZOATE
     Found Molcule (9) 1,3-DIACETOXY-BUTANE
     Found Molcule (10) METHYL 3-BENZOYLOXYPROPANOATE
     Found Molcule (11) 1-(1-HYDROXYETHYL)NAPHTHALENE
     Found Molcule (12) 2-(1-HYDROXYETHYL)NAPHTHALENE
     Found Molcule (13) N-(PHENYLETHYL)ACETAMIDE
     Found Molcule (14) 1,3 BUTANEDIOL DIBENZOATE
     Found Molcule (15) 1,4 PENTYL DIBENZOATE
     Found Molcule (16) 1,5 HEXYL DIBENZOATE
     Found Molcule (17) 3 PHENYL BUTYL BENZOATE
     Found Molcule (18) 1 METHYL 3 PHENYL PROPYL BENZOATE
     Found Molcule (19) METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE
     Found Molcule (20) METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
     Found Molcule (21) PENTYNYL 3 BENZOATE
     Found Molcule (22) BUTYNYL 3 BENZOATE
=== Found a total of 22 molecules for column Chiralcel OB.
==========================================================


************************************************************
*** Looking for molecules with column Chiralcel OD ....
     Found Molcule (1) MANDELIC ACID
     Found Molcule (2) 2-PHENYLPROPIONIC ACID
     Found Molcule (3) 2 BUTYL PHENYL CARBAMATE
     Found Molcule (4) PRACTOLOL
     Found Molcule (5) BENZOIN
     Found Molcule (6) CARTEOLOL
     Found Molcule (7) CYCLOPENTOLATE
     Found Molcule (8) GUALFENESINE
     Found Molcule (9) 1-(1-HYDROXYETHYL)NAPHTHALENE
     Found Molcule (10) 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
     Found Molcule (11) CARBINOXAMINE
     Found Molcule (12) SULCONAZOL
     Found Molcule (13) BENZYL PHENYL CARBINOL
     Found Molcule (14) PROPRANOLOL
     Found Molcule (15) PINDOLOL
     Found Molcule (16) METOPROLOL
     Found Molcule (17) ALPRENOLOL
     Found Molcule (18) PERISOXAL
     Found Molcule (19) N-CBZ-ALANINE
     Found Molcule (20) N-CBZ-NOR-LEUCINE
     Found Molcule (21) N-CBZ-VALINE
     Found Molcule (22) N-CBZ-ASPARAGINE
     Found Molcule (23) 2PHENOXYPROPIONIC ACID
=== Found a total of 23 molecules for column Chiralcel OD.
==========================================================


************************************************************
*** Looking for molecules with column Chiralpak WH ....
     Found Molcule (1) MANDELIC ACID
     Found Molcule (2) D-PHENYLALANINE
     Found Molcule (3) D-(+)-TRYPTOPHAN
     Found Molcule (4) HISTIDINE
     Found Molcule (5) L-METHIONINE
     Found Molcule (6) DL VALINE
     Found Molcule (7) L-ISOLEUCINE
     Found Molcule (8) ASPARTIC ACID
     Found Molcule (9) LACTIC ACID
     Found Molcule (10) GLYCERIC ACID
     Found Molcule (11) WARFARINE
     Found Molcule (12) N-AMIDE-L-PHENYLALANINE
     Found Molcule (13) N-ACETYL TRYPTOPHAN
     Found Molcule (14) N-ACTEYL VALINE
     Found Molcule (15) N-ACETYL-L-PHENYLALANINE
     Found Molcule (16) N-CBZ-ALANINE
     Found Molcule (17) N-CBZ NOR LEUCINE
     Found Molcule (18) N-CBZ-VALINE
     Found Molcule (19) N-CBZ ASPARAGINE
     Found Molcule (20) N-CBZ-PHENYLALANINE
     Found Molcule (21) 2-AMINO-3-PHENYLPROPIONAMIDE
     Found Molcule (22) THREONINE
     Found Molcule (23) SERINE
     Found Molcule (24) ALANINE
=== Found a total of 24 molecules for column Chiralpak WH.
==========================================================


************************************************************
*** Looking for molecules with column Chiralcel CA-1 ....
     Found Molcule (1) MANDELAMIDE
     Found Molcule (2) 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
=== Found a total of 2 molecules for column Chiralcel CA-1.
==========================================================


************************************************************
*** Looking for molecules with column Chiralcel OJ ....
     Found Molcule (1) METHYL2-PHENYLPROPIONATE
     Found Molcule (2) CYCLOPENTOLATE
     Found Molcule (3) 1-(1-HYDROXYETHYL)NAPHTHALENE
     Found Molcule (4) 2-(1-HYDROXYETHYL)NAPHTHALENE
     Found Molcule (5) SULCONAZOL
     Found Molcule (6) KETOPROFEN
     Found Molcule (7) FENOPROFEN
     Found Molcule (8) ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE
     Found Molcule (9) TRIADIMEFON
     Found Molcule (10) TIAPROFENIC ACID ME ESTER
     Found Molcule (11) METHYL 2 PHENOXYPROPIONATE
=== Found a total of 11 molecules for column Chiralcel OJ.
==========================================================


************************************************************
*** Looking for molecules with column Chiralcel OK ....
     Found Molcule (1) METHYL MANDELATE
     Found Molcule (2) 2-(1-HYDROXYETHYL)NAPHTHALENE
     Found Molcule (3) METHYL 2 PHENOXYPROPIONATE
=== Found a total of 3 molecules for column Chiralcel OK.
==========================================================


************************************************************
*** Looking for molecules with column Chiralcel OC ....
     Found Molcule (1) 1-(1-HYDROXYETHYL)NAPHTHALENE
=== Found a total of 1 molecules for column Chiralcel OC.
==========================================================


************************************************************
*** Looking for molecules with column Chiralcel OF ....
```

```
     Found Molecule (1) BENZOIN
     Found Molecule (2) PERISOXAL
     Found Molecule (3) ACETYLPHENETHOLIDE
     Found Molecule (4) 2PHENOXYPROPIONIC ACID
=== Found a total of 4 molecules for column Chiralcel OF.
==========================================================


************************************************************
*** Looking for molecules with column Chiralcel OG ....
     Found Molecule (1) 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
     Found Molecule (2) ACETYLPHENETHOLIDE
--- Found a total of 2 molecules for column Chiralcel OG.
----------------------------------------------------------


************************************************************
*** Looking for molecules with column Chiralpak WM ....
     Found Molecule (1) D-PHENYLALANINE
     Found Molecule (2) L-METHIONINE
     Found Molecule (3) LYSINE
     Found Molecule (4) DL-VALINE
     Found Molecule (5) L-ISOLEUCINE
     Found Molecule (6) L-ORNITHINE
     Found Molecule (7) ARGININE
     Found Molecule (8) ASPARTIC ACID
     Found Molecule (9) DL-LEUCINE
     Found Molecule (10) ALANINE
--- Found a total of 10 molecules for column Chiralpak WM.
==========================================================


************************************************************
*** Looking for molecules with column Chiralpak WE ....
     Found Molecule (1) L-METHIONINE
     Found Molecule (2) LYSINE
     Found Molecule (3) DL-VALINE
     Found Molecule (4) L-ISOLEUCINE
     Found Molecule (5) ARGININE
     Found Molecule (6) DL-LEUCINE
--- Found a total of 6 molecules for column Chiralpak WE.
==========================================================


************************************************************
*** Looking for molecules with column Chiralpak OP ....
     Found Molecule (1) BENZOIN
=== Found a total of 1 molecules for column Chiralpak OP.
==========================================================


************************************************************
*** Looking for molecules with column Chiralpak OT ....
     Found Molecule (1) 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
     Found Molecule (2) 1-BROMO ETHYLBENZENE
     Found Molecule (3) ETHYL 2,3 DIPHENYL PROPANOATE
     Found Molecule (4) 2 ETHYL PHENYL BENZOATE
=== Found a total of 4 molecules for column Chiralpak OT.
==========================================================


   YES

************************************************************
*** Testing all columns for predicate test_col .....

************************************************************
*** Testing for suggestions for column Crownpak CR ....
 = Ordered column suggestions [column] for molecule PHENYLGLYCINE....
+ Chiralpak WM [11], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [11], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [11], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [11], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [133], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [141], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [141], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [141], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [141], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [156], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [156], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [156], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WE [192], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [192], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [199], matching DL VALINE {Daicel p33}.
+ Chiralpak WH [199], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [199], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [199], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [286], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [344], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralpak WH [344], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [344], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [355], matching LYSINE {Daicel p38}.
+ Chiralpak WM [355], matching LYSINE {Daicel p38}.
+ Chiralpak WE [355], matching LYSINE {Daicel p38}.
+ Crownpak CR [386], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [386], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [389], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [443], matching ALANINE {Daicel p33}.
+ Chiralpak WH [443], matching ALANINE {Daicel p33}.
+ Chiralpak WM [443], matching ALANINE {Daicel p33}.
+ Chiralcel OB [476], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [489], matching DL-TYROSINE {Daicel p35}.
+ Chiralcel OD [524], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Crownpak CR [559], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
       {Daicel p35}.
+ Chiralcel OB [566], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Crownpak CR [568], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [586], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [586], matching HISTIDINE {Daicel p38}.
+ Chiralcel OK [621], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [626], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralcel CA-1 [669], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [678], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [745], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [746], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OH [822], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OH [838], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [879], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [879], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OH [889], matching 1.3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [896], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [936], matching CITRULLINE {Daicel p37}.
+ Chiralpak OT [1069], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OD [1165], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1165], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1165], matching BENZOIN {Daicel p44}.
+ Chiralcel OJ [1165], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1165], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1165], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OH [1165], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1165], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1165], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1165], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralpak OT [1175], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [1217], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OD [1232], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OJ [1329], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1329], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
```

+ Chiralcel OR [1330], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [1408], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1429], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1429], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1463], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OR [1470], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1472], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OR [1488], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OR [1547], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OD [1553], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1553], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1553], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1553], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak WH [1568], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1568], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralpak WH [1583], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralpak WH [1626], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1626], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [1647], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1647], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OR [1704], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OD [1784], matching METOPROLOL {Daicel p20}.
+ Chiralcel OR [1845], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OR [1991], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OR [2141], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2237], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OD [2358], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2377], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule D-PHENYLALANINE....
+ Chiralpak WM [112], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [112], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [112], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [112], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [156], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WE [191], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [191], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [191], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [191], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [191], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [191], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [304], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [325], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [329], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [336], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [351], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Crownpak CR [368], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [368], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [368], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [368], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [399], matching LYSINE {Daicel p38}.
+ Chiralpak WM [399], matching LYSINE {Daicel p38}.
+ Chiralpak WE [399], matching LYSINE {Daicel p38}.
+ Crownpak CR [544], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [573], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [573], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [587], matching ALANINE {Daicel p33}.
+ Chiralpak WH [587], matching ALANINE {Daicel p33}.
+ Chiralpak WM [587], matching ALANINE {Daicel p33}.
+ Chiralcel OB [617], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [695], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OK [776], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [824], matching L 3,4 DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Crownpak CR [900], matching ERYTHRO-2 AMINO-1,2 DIPHENYLETHANOL {Daicel p42}.
+ Chiralcel OR [901], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [961], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [976], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [976], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OR [977], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OJ [1012], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OR [1012], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1012], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1012], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1012], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1012], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1012], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Crownpak CR [1020], matching ARGININE {Daicel p39}.
+ Chiralpak WE [1020], matching ARGININE {Daicel p39}.
+ Chiralpak WM [1020], matching ARGININE {Daicel p39}.
+ Chiralcel OJ [1064], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OD [1068], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OR [1071], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OD [1078], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1078], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1078], matching BENZOIN {Daicel p44}.
+ Crownpak CR [1122], matching CITRULLINE {Daicel p37}.
+ Chiralcel OB [1175], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [1263], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1285], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1285], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1310], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OR [1315], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1319], matching PROPRANOLOL {Daicel p20}.
+ Chiralpak OT [1330], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OD [1400], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1400], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1400], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1400], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OJ [1426], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1426], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralpak WH [1428], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralpak WH [1503], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1503], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OR [1518], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1618], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1618], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1640], matching METOPROLOL {Daicel p20}.
+ Chiralcel OR [1693], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1694], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [1701], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [1722], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1722], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OR [1847], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1997], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2401], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OD [2437], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2543], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule 2-AMINO-2-PHENYLETHANOL....
+ Crownpak CR [133], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [141], matching 2-AMINO-1-HEXANOL {Daicel p32}.

+ Chiralpak WM [143], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [143], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [143], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [143], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [199], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [286], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [286], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [286], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [286], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [304], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [304], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [304], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [344], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [344], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [344], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [344], matching DL-VALINE {Daicel p33}.
+ Chiralcel OB [344], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [366], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [434], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OB [439], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralpak WH [476], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [476], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [500], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Crownpak CR [507], matching LYSINE {Daicel p38}.
+ Chiralpak WM [507], matching LYSINE {Daicel p38}.
+ Chiralpak WE [507], matching LYSINE {Daicel p38}.
+ Chiralcel OB [524], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [532], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [532], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel CA-1 [578], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OK [598], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [632], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Crownpak CR [638], matching DL-TYROSINE {Daicel p35}.
+ Chiralcel OD [678], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Crownpak CR [713], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OB [723], matching 1-PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [727], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [727], matching HISTIDINE {Daicel p38}.
+ Chiralcel OB [798], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [906], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralpak OT [932], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OD [1011], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1011], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Crownpak CR [1077], matching CITRULLINE {Daicel p37}.
+ Chiralcel OD [1141], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1141], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1141], matching BENZOIN {Daicel p44}.
+ Chiralcel OJ [1141], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1141], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1141], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1141], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1141], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1141], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1141], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1209], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OD [1416], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1416], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1440], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1448], matching PROPRANOLOL {Daicel p28}.
+ Chiralcel OJ [1471], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1471], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1529], matching 9 (2 TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1529], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1529], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1529], matching 9 (2 TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OD [1600], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OB [1753], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [1983], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OB [2003], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OD [2326], matching CARTEOLOL {Daicel p20}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule 2-AMINO-1-PHENYLETHANOL....
+ Chiralcel OK [209], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [269], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel CA-1 [273], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [288], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [366], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [389], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [400], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [400], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [400], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [400], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [406], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [406], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [433], matching 1-PHENYL PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [463], matching 1-PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [544], matching 2 AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [544], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [544], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [544], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [575], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [575], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [575], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [575], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [596], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Crownpak CR [602], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralpak WE [627], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [627], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [634], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [634], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [634], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [634], matching DL-VALINE {Daicel p33}.
+ Chiralcel OB [672], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OD [704], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralpak WH [775], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [790], matching LYSINE {Daicel p38}.
+ Chiralpak WM [790], matching LYSINE {Daicel p38}.
+ Chiralpak WE [790], matching LYSINE {Daicel p38}.
+ Chiralcel OD [790], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [790], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [790], matching BENZOIN {Daicel p44}.
+ Crownpak CR [821], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [821], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OJ [834], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OD [857], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Crownpak CR [877], matching ALANINE {Daicel p33}.
+ Chiralpak WH [877], matching ALANINE {Daicel p33}.
+ Chiralpak WM [877], matching ALANINE {Daicel p33}.
+ Chiralcel OB [882], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [882], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [882], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [882], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [882], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [882], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [882], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [950], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [958], matching DL-TYROSINE {Daicel p35}.
+ Chiralpak OT [974], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OB [1020], matching 2-ACETOXY 3 PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [1051], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1051], matching PERISOXAL {Daicel p20}.

+ Chiralcel OD [1075], matching ALPRENOLOL (Daicel p20).
+ Crownpak CR [1078], matching ERYTHRO 2 AMINO 1,2-DIPHENYLETHANOL (Daicel p32).
+ Chiralcel OD [1084], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OB [1110], matching 1,3-DIACETOXY BUTANE (Daicel p45).
+ Chiralcel OJ [1167], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OK [1167], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OD [1303], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [1303], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [1303], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel CA-1 [1303], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Crownpak CR [1336], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [1336], matching HISTIDINE (Daicel p38).
+ Chiralcel OD [1350], matching METOPROLOL (Daicel p20).
+ Crownpak CR [1359], matching L-GLUTAMIC ACID (Daicel p36).
+ Chiralcel OD [1364], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [1364], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Chiralcel OJ [1380], matching FENOPROFEN (Daicel p27).
+ Chiralcel OJ [1522], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OD [1522], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OJ [1571], matching KETOPROFEN (Daicel p27).
+ Chiralpak WH [1614], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1614], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralpak WH [1629], matching N-CBZ-PHENYLALANINE (Daicel p40).
+ Chiralcel OD [1638], matching CARBINOXAMINE (Daicel p21).
+ Chiralpak WH [1673], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1673], matching N-CBZ-VALINE (Daicel p40).
+ Crownpak CR [1686], matching CITRULLINE (Daicel p37).
+ Chiralpak WH [1811], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [1811], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OJ [1893], matching TRIADIMEFON (Daicel p22).
+ Chiralcel OD [1924], matching CARTEOLOL (Daicel p20).
+ Chiralcel OB [2037], matching N,O-DIBENZOYL2-AMINOBUTANOL (Daicel p46).
+ Chiralcel OD [2114], matching 2 BUTYL PHENYL CARBAMATE (Daicel p46).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule 1-PHENYLETHYLAMINE....
+ Chiralcel OB [339], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OB [344], matching 1-PHENYLETHANOL (Daicel p42).
+ Crownpak CR [367], matching 2 AMINO BUTANE (Daicel p32).
+ Chiralcel OJ [415], matching METHYL2-PHENYLPROPIONATE (Daicel p28).
+ Chiralcel OB [430], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OD [472], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralpak OT [479], matching 1-BROMO ETHYLBENZENE (Daicel p46).
+ Crownpak CR [500], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [626], matching PHENYLGLYCINE (Daicel p34).
+ Chiralpak WM [637], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [637], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [637], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [637], matching L-METHIONINE (Daicel p36).
+ Chiralcel OB [651], matching 2-ACETOXY-3-PHENYLPROPANE (Daicel p45).
+ Chiralcel OK [656], matching METHYL MANDELATE (Daicel p44).
+ Crownpak CR [790], matching 2-AMINO-3-METHYLBUTANE (Daicel p32).
+ Chiralcel OD [805], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [805], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Chiralcel OB [838], matching 1-PHENYL-1,3 PROPANEDIOL (Daicel p44).
+ Chiralcel OB [881], matching 1 PHENYL-1,2 ETHANEDIOL (Daicel p44).
+ Crownpak CR [981], matching LYSINE (Daicel p38).
+ Chiralpak WM [981], matching LYSINE (Daicel p38).
+ Chiralpak WE [981], matching LYSINE (Daicel p38).
+ Chiralpak WH [1008], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [1008], matching MANDELIC ACID (Daicel p27).
+ Chiralpak OT [1159], matching ETHYL 2,3 DIPHENYL PROPANOATE (Daicel p46).
+ Chiralcel OJ [1196], matching TIAPROFENIC ACID ME ESTER (Daicel p28).
+ Chiralcel OB [1197], matching 3 PHENYL BUTYL BENZOATE (Daicel p45).
+ Crownpak CR [1212], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [1212], matching HISTIDINE (Daicel p38).
+ Crownpak CR [1562], matching CITRULLINE (Daicel p37).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL....
+ Crownpak CR [632], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Chiralcel OD [663], matching BENZOIN (Daicel p44).
+ Chiralcel OF [663], matching BENZOIN (Daicel p44).
+ Chiralpak OP [663], matching BENZOIN (Daicel p44).
+ Chiralpak OT [700], matching 2 ETHYL PHENYL BENZOATE (Daicel p46).
+ Chiralcel OD [735], matching BENZYL PHENYL CARBINOL (Daicel p46).
+ Crownpak CR [745], matching PHENYLGLYCINE (Daicel p34).
+ Chiralpak WM [802], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [802], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [802], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [802], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [821], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Crownpak CR [856], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Crownpak CR [900], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [900], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [900], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [954], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Crownpak CR [1089], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralcel OD [1098], matching CARBINOXAMINE (Daicel p21).
+ Chiralcel OJ [1169], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OD [1169], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OB [1172], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Crownpak CR [1234], matching DL-TYROSINE (Daicel p35).
+ Chiralcel OB [1252], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
+ Chiralcel OD [1271], matching 2 BUTYL PHENYL CARBAMATE (Daicel p46).
+ Chiralcel OK [1310], matching METHYL MANDELATE (Daicel p44).
+ Chiralcel OB [1368], matching N,O-DIBENZOYL2-AMINOBUTANOL (Daicel p46).
+ Chiralcel OB [1435], matching 1-PHENYLETHANOL (Daicel p42).
+ Chiralcel OB [1511], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OJ [1894], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1894], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OK [1894], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1894], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OC [1894], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [1894], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [1894], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [2176], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [2176], matching PERISOXAL (Daicel p20).
+ Chiralcel OD [2235], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OD [2253], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OD [2300], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [2300], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [2300], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel CA-1 [2300], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OD [2438], matching METOPROLOL (Daicel p20).
+ Chiralcel OD [3022], matching CARTEOLOL (Daicel p20).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule 2-AMINO-3-METHYLBUTANE....
+ Crownpak CR [407], matching 2 AMINO BUTANE (Daicel p32).
+ Chiralcel OJ [598], matching METHYL2-PHENYLPROPIONATE (Daicel p28).
+ Chiralpak OT [633], matching 1-BROMO ETHYLBENZENE (Daicel p46).
+ Crownpak CR [774], matching 1-PHENYLETHYLAMINE (Daicel p32).
+ Chiralcel OB [1328], matching 3 PHENYL BUTYL BENZOATE (Daicel p45).
+ Chiralcel OJ [1378], matching TIAPROFENIC ACID ME ESTER (Daicel p28).

```
************************************************************
*** Testing for suggestions for column Crownpak CR ....
 - Ordered column suggestions [column] for molecule 2-AMINO-1-HEXANOL....
 + Crownpak CR [70], matching 2-AMINO-1-PENTANOL [Daicel p32].
 + Crownpak CR [133], matching L-ISOLEUCINE [Daicel p34].
 + Chiralpak WH [133], matching L-ISOLEUCINE [Daicel p34].
 + Chiralpak WM [133], matching L-ISOLEUCINE [Daicel p34].
 + Chiralpak WE [133], matching L-ISOLEUCINE [Daicel p34].
 + Chiralpak WE [133], matching DL-LEUCINE [Daicel p33].
 + Chiralpak WM [133], matching DL-LEUCINE [Daicel p33].
 + Crownpak CR [141], matching 2-AMINO-2-PHENYLETHANOL [Daicel p32].
 + Crownpak CR [206], matching DL-VALINE [Daicel p33].
 + Chiralpak WH [206], matching DL-VALINE [Daicel p33].
 + Chiralpak WM [206], matching DL-VALINE [Daicel p33].
 + Chiralpak WE [206], matching DL-VALINE [Daicel p33].
 + Chiralpak WM [266], matching L-METHIONINE [Daicel p36].
 + Crownpak CR [266], matching L-METHIONINE [Daicel p36].
 + Chiralpak WH [266], matching L-METHIONINE [Daicel p36].
 + Chiralpak WE [266], matching L-METHIONINE [Daicel p36].
 + Crownpak CR [286], matching PHENYLGLYCINE [Daicel p44].
 + Crownpak CR [336], matching D-PHENYLALANINE [Daicel p34].
 + Chiralpak WH [336], matching D-PHENYLALANINE [Daicel p34].
 + Chiralpak WM [336], matching D-PHENYLALANINE [Daicel p34].
 + Chiralpak WH [449], matching 2-AMINO-3-PHENYLPROPIONAMIDE [Daicel p41].
 + Crownpak CR [488], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
       [Daicel p35].
 + Crownpak CR [501], matching LYSINE [Daicel p38].
 + Chiralpak WM [501], matching LYSINE [Daicel p38].
 + Chiralpak WE [501], matching LYSINE [Daicel p38].
 + Crownpak CR [504], matching L-ORNITHINE [Daicel p38].
 + Chiralpak WM [504], matching L-ORNITHINE [Daicel p38].
 + Chiralcel OB [507], matching 1-PHENYL-1,2-ETHANEDIOL [Daicel p44].
 + Crownpak CR [510], matching ALANINE [Daicel p33].
 + Chiralpak WH [510], matching ALANINE [Daicel p33].
 + Chiralpak WM [510], matching ALANINE [Daicel p33].
 + Crownpak CR [544], matching 2-AMINO-1-PHENYLETHANOL [Daicel p32].
 + Chiralcel OB [592], matching 1-PHENYL-1,3-PROPANEDIOL [Daicel p44].
 + Chiralpak WH [652], matching MANDELIC ACID [Daicel p27].
 + Chiralcel OD [652], matching MANDELIC ACID [Daicel p27].
 + Chiralcel OB [681], matching 2-PHENYL-PROPAN-1-OL [Daicel p42].
 + Crownpak CR [746], matching DL-TYROSINE [Daicel p35].
 + Chiralcel OK [765], matching METHYL MANDELATE [Daicel p44].
 + Chiralcel CA-1 [766], matching MANDELAMIDE [Daicel p44].
 + Crownpak CR [788], matching L-GLUTAMIC ACID [Daicel p36].
 + Crownpak CR [826], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL [Daicel p32].
 + Chiralcel OB [863], matching 1-PHENYLETHANOL [Daicel p42].
 + Chiralcel OB [939], matching 1-PHENYL-PROPAN-1-OL [Daicel p42].
 + Crownpak CR [988], matching L-GLUTAMINE [Daicel p36].
 + Chiralcel OB [1046], matching N-(PHENYLETHYL)ACETAMIDE [Daicel p46].
 + Chiralcel OB [1132], matching 1,3-DIACETOXY-BUTANE [Daicel p45].
 + Crownpak CR [1193], matching L-3,4-DIHYDROXYPHENYLALANINE [Daicel p35].
 + Chiralcel OD [1262], matching BENZYL PHENYL CARBINOL [Daicel p46].
 + Chiralcel OJ [1263], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OB [1263], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OK [1263], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OB [1263], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OC [1263], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OJ [1263], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OD [1263], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OD [1345], matching BENZOIN [Daicel p44].
 + Chiralcel OF [1345], matching BENZOIN [Daicel p44].
 + Chiralpak OP [1345], matching BENZOIN [Daicel p44].
 + Chiralcel OD [1548], matching CARBINOXAMINE [Daicel p21].
 + Chiralcel OD [1561], matching PERISOXAL [Daicel p20].
 + Chiralcel OF [1561], matching PERISOXAL [Daicel p20].
 + Chiralcel OB [1611], matching METHYL-3-BENZOYLOXYPROPANOATE [Daicel p45].
 + Chiralcel OD [1630], matching PROPRANOLOL [Daicel p20].
 + Chiralcel OD [1648], matching ALPRENOLOL [Daicel p20].
 + Chiralcel OD [1669], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
       [Daicel p43].
 + Chiralcel OG [1669], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
       [Daicel p43].
 + Chiralpak OT [1669], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
       [Daicel p43].
 + Chiralcel CA-1 [1669], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
       [Daicel p43].
 + Chiralcel OD [1834], matching METOPROLOL [Daicel p20].
 + Chiralcel OD [1902], matching 2 BUTYL PHENYL CARBAMATE [Daicel p46].
 + Chiralcel OB [2084], matching N,O-DIBENZOYL2-AMINOBUTANOL [Daicel p46].
 + Chiralcel OJ [2130], matching CYCLOPENTOLATE [Daicel p21].
 + Chiralcel OD [2130], matching CYCLOPENTOLATE [Daicel p21].
 + Chiralcel OD [2307], matching CARTEOLOL [Daicel p20].

************************************************************
*** Testing for suggestions for column Crownpak CR ....
 - Ordered column suggestions [column] for molecule 2-AMINO-1-PENTANOL....
 + Crownpak CR [70], matching 2-AMINO-1-HEXANOL [Daicel p32].
 + Crownpak CR [133], matching DL-VALINE [Daicel p33].
 + Chiralpak WH [133], matching DL-VALINE [Daicel p33].
 + Chiralpak WM [133], matching DL-VALINE [Daicel p33].
 + Chiralpak WE [133], matching DL-VALINE [Daicel p33].
 + Crownpak CR [199], matching 2-AMINO-2-PHENYLETHANOL [Daicel p32].
 + Crownpak CR [206], matching L-ISOLEUCINE [Daicel p34].
 + Chiralpak WH [206], matching L-ISOLEUCINE [Daicel p34].
 + Chiralpak WM [206], matching L-ISOLEUCINE [Daicel p34].
 + Chiralpak WE [206], matching L-ISOLEUCINE [Daicel p34].
 + Chiralpak WE [294], matching DL-LEUCINE [Daicel p33].
 + Chiralpak WM [294], matching DL-LEUCINE [Daicel p33].
 + Crownpak CR [344], matching PHENYLGLYCINE [Daicel p44].
 + Crownpak CR [368], matching ALANINE [Daicel p33].
 + Chiralpak WH [368], matching ALANINE [Daicel p33].
 + Chiralpak WM [368], matching ALANINE [Daicel p33].
 + Chiralpak WM [368], matching L-METHIONINE [Daicel p36].
 + Crownpak CR [368], matching L-METHIONINE [Daicel p36].
 + Chiralpak WH [368], matching L-METHIONINE [Daicel p36].
 + Chiralpak WE [368], matching L-METHIONINE [Daicel p36].
 + Crownpak CR [499], matching L-ORNITHINE [Daicel p38].
 + Chiralpak WM [499], matching L-ORNITHINE [Daicel p38].
 + Chiralcel OB [566], matching 1-PHENYL-1,2-ETHANEDIOL [Daicel p44].
 + Crownpak CR [602], matching 2-AMINO-1-PHENYLETHANOL [Daicel p32].
 + Chiralpak WH [647], matching 2-AMINO-3-PHENYLPROPIONAMIDE [Daicel p41].
 + Chiralcel OB [650], matching 1-PHENYL-1,3-PROPANEDIOL [Daicel p44].
 + Crownpak CR [659], matching L-GLUTAMIC ACID [Daicel p36].
 + Crownpak CR [666], matching THREONINE [Daicel p35].
 + Chiralpak WH [666], matching THREONINE [Daicel p35].
 + Chiralpak WH [711], matching MANDELIC ACID [Daicel p27].
 + Chiralcel OD [711], matching MANDELIC ACID [Daicel p27].
 + Crownpak CR [722], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
       [Daicel p35].
 + Chiralcel OB [739], matching 2-PHENYL-PROPAN-1-OL [Daicel p42].
 + Crownpak CR [786], matching ASPARTIC ACID [Daicel p37].
 + Chiralpak WH [786], matching ASPARTIC ACID [Daicel p37].
 + Chiralpak WM [786], matching ASPARTIC ACID [Daicel p37].
 + Chiralcel OK [823], matching METHYL MANDELATE [Daicel p44].
 + Chiralcel CA-1 [824], matching MANDELAMIDE [Daicel p44].
 + Crownpak CR [885], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL [Daicel p32].
 + Crownpak CR [911], matching L-GLUTAMINE [Daicel p36].
 + Crownpak CR [912], matching HISTIDINE [Daicel p38].
 + Chiralpak WH [912], matching HISTIDINE [Daicel p38].
 + Chiralcel OB [921], matching 1-PHENYLETHANOL [Daicel p42].
 + Chiralcel OB [990], matching 1,3-DIACETOXY BUTANE [Daicel p45].
 + Chiralcel OB [997], matching 1-PHENYL-PROPAN-1-OL [Daicel p42].
 + Crownpak CR [1017], matching ASPARAGINE [Daicel p37].
 + Chiralcel OB [1105], matching N-(PHENYLETHYL)ACETAMIDE [Daicel p46].
 + Crownpak CR [1297], matching CITRULLINE [Daicel p37].
 + Chiralcel OB [1405], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OC [1405], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
 + Chiralcel OJ [1405], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
```

+ Chiralcel OD [1405], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [1405], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1405], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OK [1405], matching 2-(1 HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [1439], matching BENZYL PHENYL CARBINOL (Daicel p46).
+ Chiralcel OB [1469], matching METHYL-3-BENZOYLOXYPROPANOATE (Daicel p45).
+ Chiralcel OD [1508], matching BENZOIN (Daicel p44).
+ Chiralcel OF [1508], matching BENZOIN (Daicel p44).
+ Chiralpak OP [1508], matching BENZOIN (Daicel p44).
+ Chiralcel OD [1703], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [1703], matching PERISOXAL (Daicel p20).
+ Chiralcel OD [1704], matching CARBINOXAMINE (Daicel p21).
+ Chiralcel OD [1760], matching 2 BUTYL PHENYL CARBAMATE (Daicel p46).
+ Chiralcel OD [1772], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OD [1791], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OD [1811], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [1811], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [1811], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel CA-1 [1811], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OD [1976], matching METOPROLOL (Daicel p20).
+ Chiralcel OB [2226], matching N,O DIBENZOYL2-AMINOBUTANOL (Daicel p46).
+ Chiralcel OJ [2273], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OD [2273], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OD [2311], matching CARTEOLOL (Daicel p20).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule DL-TYROSINE....
+ Crownpak CR [329], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [329], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [329], matching D-PHENYLALANINE (Daicel p34).
+ Crownpak CR [431], matching L-3,4-DIHYDROXYPHENYLALANINE (Daicel p35).
+ Chiralpak WM [444], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [444], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [444], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [444], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [474], matching L-GLUTAMIC ACID (Daicel p36).
+ Crownpak CR [489], matching PHENYLGLYCINE (Daicel p34).
+ Crownpak CR [589], matching ARGININE (Daicel p39).
+ Chiralpak WE [589], matching ARGININE (Daicel p39).
+ Chiralpak WM [589], matching ARGININE (Daicel p39).
+ Chiralpak WE [600], matching DL-LEUCINE (Daicel p33).
+ Chiralpak WM [600], matching DL-LEUCINE (Daicel p33).
+ Crownpak CR [601], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WH [601], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [601], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [601], matching L-ISOLEUCINE (Daicel p34).
+ Chiralcel OD [615], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [615], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Crownpak CR [628], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WH [628], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [628], matching ASPARTIC ACID (Daicel p37).
+ Crownpak CR [638], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [645], matching LYSINE (Daicel p38).
+ Chiralpak WM [645], matching LYSINE (Daicel p38).
+ Chiralpak WE [645], matching LYSINE (Daicel p38).
+ Chiralpak WH [654], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Crownpak CR [724], matching CITRULLINE (Daicel p37).
+ Crownpak CR [746], matching 2-AMINO 1 HEXANOL (Daicel p32).
+ Crownpak CR [958], matching 2-AMINO-1 PHENYLETHANOL (Daicel p32).
+ Chiralcel OB [1155], matching BUTYNYL 3 BENZOATE (Daicel p45).
+ Crownpak CR [1234], matching ERYTHRO 2-AMINO-1,2-DIPHENYLETHANOL (Daicel p32).
+ Chiralcel OJ [1323], matching TRIADIMEFON (Daicel p22).
+ Chiralcel OD [1694], matching PRACTOLOL (Daicel p20).
+ Chiralcel OD [1697], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OD [1699], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [1699], matching PERISOXAL (Daicel p20).
+ Chiralcel OD [1708], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OD [3041], matching METOPROLOL (Daicel p20).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule L-3,4-DIHYDROXYPHENYLALANINE....
+ Crownpak CR [431], matching DL-TYROSINE (Daicel p35).
+ Crownpak CR [651], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [651], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Crownpak CR [709], matching L-GLUTAMIC ACID (Daicel p36).
+ Crownpak CR [716], matching ARGININE (Daicel p39).
+ Chiralpak WE [716], matching ARGININE (Daicel p39).
+ Chiralpak WM [716], matching ARGININE (Daicel p39).
+ Crownpak CR [824], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [824], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [824], matching D-PHENYLALANINE (Daicel p34).
+ Crownpak CR [845], matching LYSINE (Daicel p38).
+ Chiralpak WM [845], matching LYSINE (Daicel p38).
+ Chiralpak WE [845], matching LYSINE (Daicel p38).
+ Crownpak CR [873], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WH [873], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [873], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [895], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [895], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [895], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [895], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [962], matching THREONINE (Daicel p35).
+ Chiralpak WH [962], matching THREONINE (Daicel p35).
+ Chiralpak WE [1048], matching DL-LEUCINE (Daicel p33).
+ Chiralpak WM [1048], matching DL-LEUCINE (Daicel p33).
+ Crownpak CR [1048], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WH [1048], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [1048], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [1048], matching L-ISOLEUCINE (Daicel p34).
+ Crownpak CR [1111], matching SERINE (Daicel p35).
+ Chiralpak WH [1111], matching SERINE (Daicel p35).
+ Chiralpak WH [1148], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Crownpak CR [1193], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Chiralcel OB [1380], matching BUTYNYL 3 BENZOATE (Daicel p45).
+ Chiralcel OD [1972], matching 2 BUTYL PHENYL CARBAMATE (Daicel p46).
+ Chiralcel OD [2079], matching METOPROLOL (Daicel p20).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule D-(+)-TRYPTOPHAN....
+ Crownpak CR [631], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [631], matching HISTIDINE (Daicel p38).
+ Crownpak CR [643], matching L-GLUTAMIC ACID (Daicel p36).
+ Crownpak CR [651], matching L-3,4-DIHYDROXYPHENYLALANINE (Daicel p35).
+ Chiralcel OD [668], matching GUAIFENESINE (Daicel p44).
+ Crownpak CR [707], matching THREONINE (Daicel p35).
+ Chiralpak WH [707], matching THREONINE (Daicel p35).
+ Crownpak CR [781], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WH [781], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [781], matching ASPARTIC ACID (Daicel p37).
+ Crownpak CR [821], matching L-GLUTAMINE (Daicel p36).
+ Crownpak CR [861], matching SERINE (Daicel p35).
+ Chiralpak WH [861], matching SERINE (Daicel p35).
+ Crownpak CR [983], matching ASPARAGINE (Daicel p37).
+ Chiralcel OD [1059], matching PINDOLOL (Daicel p20).
+ Chiralpak WH [1272], matching N-CBZ ALANINE (Daicel p40).
+ Chiralcel OD [1272], matching N-CBZ ALANINE (Daicel p40).
+ Chiralcel OJ [1314], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1314], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OK [1314], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1314], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).

+ Chiralcel OC [1314], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [1314], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [1314], matching 1-(1 HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [1429], matching METOPROLOL (Daicel p20).
+ Chiralpak WH [1582], matching LACTIC ACID (Daicel p27).
+ Chiralpak WH [1742], matching N-ACTEYL VALINE (Daicel p39).
+ Chiralcel OB [1775], matching 1 PHENYL-1,3-PROPANEDIOL (Daicel p44).
+ Chiralpak WH [1904], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [1904], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OB [1929], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Chiralpak WH [2009], matching N-ACETYL-L-PHENYLALANINE (Daicel p39).
+ Chiralcel CA-1 [2163], matching MANDELAMIDE (Daicel p44).
+ Chiralcel OJ [2334], matching TRIADIMEFON (Daicel p22).
+ Chiralpak WH [2410], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [2410], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [3021], matching PRACTOLOL (Daicel p20).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule L-GLUTAMINE....
+ Crownpak CR [171], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [171], matching HISTIDINE (Daicel p38).
+ Crownpak CR [173], matching ASPARAGINE (Daicel p37).
+ Crownpak CR [317], matching L-GLUTAMIC ACID (Daicel p36).
+ Crownpak CR [364], matching THREONINE (Daicel p35).
+ Chiralpak WH [364], matching THREONINE (Daicel p35).
+ Crownpak CR [385], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [385], matching L-ORNITHINE (Daicel p38).
+ Crownpak CR [386], matching CITRULLINE (Daicel p37).
+ Crownpak CR [386], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WH [386], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [386], matching ASPARTIC ACID (Daicel p37).
+ Crownpak CR [577], matching SERINE (Daicel p35).
+ Chiralpak WH [577], matching SERINE (Daicel p35).
+ Crownpak CR [693], matching ARGININE (Daicel p39).
+ Chiralpak WE [693], matching ARGININE (Daicel p39).
+ Chiralpak WM [693], matching ARGININE (Daicel p39).
+ Crownpak CR [759], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [759], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [759], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [759], matching DL-VALINE (Daicel p33).
+ Crownpak CR [821], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [821], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Crownpak CR [836], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WH [836], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [836], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [836], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [839], matching DL-LEUCINE (Daicel p33).
+ Chiralpak WM [839], matching DL-LEUCINE (Daicel p33).
+ Crownpak CR [857], matching ALANINE (Daicel p33).
+ Chiralpak WH [857], matching ALANINE (Daicel p33).
+ Chiralpak WM [857], matching ALANINE (Daicel p33).
+ Crownpak CR [911], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Chiralpak WH [932], matching N-ACTEYL VALINE (Daicel p39).
+ Crownpak CR [988], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Chiralpak WH [1199], matching N ACETYL-L-PHENYLALANINE (Daicel p39).
+ Chiralpak WH [1224], matching LACTIC ACID (Daicel p27).
+ Chiralcel CA-1 [1365], matching MANDELAMIDE (Daicel p44).
+ Chiralcel OB [1467], matching 1-PHENYL-1,3-PROPANEDIOL (Daicel p44).
+ Chiralpak WH [1600], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [1600], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralpak WH [1643], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [1643], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [1660], matching PINDOLOL (Daicel p20).
+ Chiralcel OB [1680], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Chiralcel OD [1712], matching PRACTOLOL (Daicel p20).
+ Chiralcel OJ [1712], matching TRIADIMEFON (Daicel p22).
+ Chiralcel OD [1945], matching GUALFENESINE (Daicel p44).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule HISTIDINE....
+ Crownpak CR [171], matching L-GLUTAMINE (Daicel p36).
+ Crownpak CR [254], matching L-GLUTAMIC ACID (Daicel p36).
+ Crownpak CR [306], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [306], matching L-ORNITHINE (Daicel p38).
+ Crownpak CR [348], matching LYSINE (Daicel p38).
+ Chiralpak WM [348], matching LYSINE (Daicel p38).
+ Chiralpak WE [348], matching LYSINE (Daicel p38).
+ Crownpak CR [353], matching CITRULLINE (Daicel p37).
+ Crownpak CR [360], matching ASPARAGINE (Daicel p37).
+ Crownpak CR [401], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WH [401], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [401], matching ASPARTIC ACID (Daicel p37).
+ Crownpak CR [434], matching THREONINE (Daicel p35).
+ Chiralpak WH [434], matching THREONINE (Daicel p35).
+ Crownpak CR [586], matching PHENYLGLYCINE (Daicel p34).
+ Crownpak CR [600], matching SERINE (Daicel p35).
+ Chiralpak WH [600], matching SERINE (Daicel p35).
+ Chiralpak WM [606], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [606], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [606], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [606], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [631], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [631], matching D-(+) TRYPTOPHAN (Daicel p38).
+ Crownpak CR [727], matching 2-AMINO-2-PHENYLETHANOL (Daicel p42).
+ Crownpak CR [767], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [767], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [767], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [767], matching DL-VALINE (Daicel p33).
+ Crownpak CR [891], matching ALANINE (Daicel p33).
+ Chiralpak WH [891], matching ALANINE (Daicel p33).
+ Chiralpak WM [891], matching ALANINE (Daicel p33).
+ Crownpak CR [912], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Chiralcel OD [929], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [929], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Chiralpak WH [930], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [930], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OB [1070], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Chiralpak WH [1118], matching N-ACTEYL VALINE (Daicel p39).
+ Chiralcel OD [1131], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralcel OB [1156], matching 1-PHENYL-1,3-PROPANEDIOL (Daicel p44).
+ Crownpak CR [1212], matching 1-PHENYLETHYLAMINE (Daicel p32).
+ Chiralcel CA-1 [1255], matching MANDELAMIDE (Daicel p44).
+ Chiralpak WH [1258], matching LACTIC ACID (Daicel p27).
+ Chiralcel OB [1272], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Crownpak CR [1336], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralpak WH [1386], matching N-ACETYL-L-PHENYLALANINE (Daicel p39).
+ Chiralcel OD [1553], matching PINDOLOL (Daicel p20).
+ Chiralcel OB [1594], matching 1-PHENYLETHANOL (Daicel p42).
+ Chiralcel OB [1609], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OD [1618], matching PRACTOLOL (Daicel p20).
+ Chiralpak OT [1654], matching 1-BROMO ETHYLBENZENE (Daicel p46).
+ Chiralcel OJ [1674], matching TRIADIMEFON (Daicel p22).
+ Chiralpak WH [1787], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [1787], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [1969], matching GUALFENESINE (Daicel p44).

****

*** Testing for suggestions for column Crownpak CR ....

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule
4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID....
+ Crownpak CR [351], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [351], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [351], matching D-PHENYLALANINE (Daicel p34).

+ Chiralpak WE [400], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [400], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [400], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [400], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [400], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [400], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [488], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OB [516], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralpak WM [521], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [521], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [521], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [521], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [541], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Crownpak CR [559], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [596], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [623], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [623], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [623], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [623], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [722], matching 2 AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [748], matching LYSINE {Daicel p38}.
+ Chiralpak WM [748], matching LYSINE {Daicel p38}.
+ Chiralpak WE [748], matching LYSINE {Daicel p38}.
+ Chiralcel OD [799], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralpak WH [819], matching N-AMIDE-L-PHENYLALANINE {Daicel p41}.
+ Chiralcel OB [830], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OK [841], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [844], matching ALANINE {Daicel p33}.
+ Chiralpak WH [844], matching ALANINE {Daicel p33}.
+ Chiralpak WM [844], matching ALANINE {Daicel p33}.
+ Chiralcel OB [965], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [967], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [967], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OB [1023], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OB [1042], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1058], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [1148], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralpak OT [1163], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OB [1181], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1216], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1216], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1216], matching BENZOIN {Daicel p44}.
+ Chiralcel OJ [1244], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [1251], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1251], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1251], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1251], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1251], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1251], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1251], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1254], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1254], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1297], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OJ [1400], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1406], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1406], matching PERISOXAL {Daicel p20}.
+ Chiralcel OB [1408], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1411], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OD [1475], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1493], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OB [1560], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [1640], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1640], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1678], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [1690], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1690], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1690], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1690], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OB [1706], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [1766], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1766], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [1842], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1842], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OB [1856], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [1995], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OB [2254], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [2334], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OD [2455], matching CARTEOLOL {Daicel p20}.

****

*** Testing for suggestions for column Crownpak CR ....
- Ordered column suggestions [column] for molecule L-METHIONINE....
+ Crownpak CR [11], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [112], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [112], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [112], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [126], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [126], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [126], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [126], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [126], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [126], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [143], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [223], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [223], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [223], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [223], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [243], matching LYSINE {Daicel p38}.
+ Chiralpak WM [243], matching LYSINE {Daicel p38}.
+ Chiralpak WE [243], matching LYSINE {Daicel p38}.
+ Crownpak CR [266], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [295], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [295], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WH [355], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [355], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [368], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [400], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [444], matching DL-TYROSINE {Daicel p35}.
+ Chiralpak WH [456], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [480], matching ALANINE {Daicel p33}.
+ Chiralpak WH [480], matching ALANINE {Daicel p33}.
+ Chiralpak WM [480], matching ALANINE {Daicel p33}.
+ Chiralcel OB [487], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [521], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralcel OD [535], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [577], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Crownpak CR [606], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [606], matching HISTIDINE {Daicel p38}.
+ Chiralcel OK [632], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [637], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralcel CA-1 [680], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [689], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [728], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [756], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [757], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [785], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [833], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [895], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Chiralcel OB [907], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralpak OT [1080], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OD [1179], matching BENZYL PHENYL CARBINOL {Daicel p46}.

+ Chiralcel OB [1181], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OC [1181], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [1181], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [1181], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [1181], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1181], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OK [1181], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralpak OT [1185], matching 2 ETHYL PHENYL BENZOATE (Daicel p46).
+ Chiralcel OD [1187], matching BENZOIN (Daicel p44).
+ Chiralcel OF [1187], matching BENZOIN (Daicel p44).
+ Chiralpak OP [1187], matching BENZOIN (Daicel p44).
+ Chiralcel OJ [1235], matching FENOPROFEN (Daicel p27).
+ Chiralcel OB [1286], matching ALPHA METHYL PHENETHYL BENZOATE (Daicel p45).
+ Chiralcel OJ [1335], matching KETOPROFEN (Daicel p27).
+ Chiralcel OJ [1380], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OK [1380], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OB [1453], matching METHYL-3-BENZOYLOXYPROPANOATE (Daicel p45).
+ Chiralcel OD [1464], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [1464], matching PERISOXAL (Daicel p20).
+ Chiralcel OD [1477], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OD [1487], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OB [1495], matching 1 METHYL 3 PHENYL PROPYL BENZOATE (Daicel p45).
+ Chiralpak WH [1539], matching N-CBZ-PHENYLALANINE (Daicel p40).
+ Chiralpak WH [1553], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1553], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1567], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [1567], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [1567], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel CA-1 [1567], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OB [1587], matching PENTYNYL 3 BENZOATE (Daicel p45).
+ Chiralcel OB [1596], matching BUTYNYL 3 BENZOATE (Daicel p45).
+ Chiralpak WH [1610], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [1610], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralpak WH [1650], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1650], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1716], matching METOPROLOL (Daicel p20).
+ Chiralcel OB [1807], matching 1,3 BUTANEDIOL DIBENZOATE (Daicel p45).
+ Chiralcel OB [1953], matching 1,4 PENTYL DIBENZOATE (Daicel p45).
+ Chiralcel OB [2077], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE (Daicel p29).
+ Chiralcel OB [2103], matching 1,5 HEXYL DIBENZOATE (Daicel p45).
+ Chiralcel OD [2265], matching CARTEOLOL (Daicel p20).
+ Chiralcel OB [2302], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE (Daicel p29).

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions (column) for molecule LYSINE....
+ Crownpak CR [144], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [144], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [243], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [243], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [243], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [243], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [348], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [348], matching HISTIDINE (Daicel p38).
+ Crownpak CR [355], matching PHENYLGLYCINE (Daicel p34).
+ Chiralpak WE [360], matching DL-LEUCINE (Daicel p33).
+ Chiralpak WM [360], matching DL-LEUCINE (Daicel p33).
+ Crownpak CR [360], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WH [360], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [360], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [360], matching L-ISOLEUCINE (Daicel p34).
+ Crownpak CR [399], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [399], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [399], matching D-PHENYLALANINE (Daicel p34).
+ Crownpak CR [501], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Crownpak CR [507], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [547], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [547], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [547], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [547], matching DL-VALINE (Daicel p33).
+ Crownpak CR [645], matching DL-TYROSINE (Daicel p35).
+ Crownpak CR [720], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID (Daicel p35).
+ Chiralpak WH [737], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [737], matching MANDELIC ACID (Daicel p27).
+ Chiralpak WH [744], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Chiralcel OB [748], matching 1,3-DIACETOXY-BUTANE (Daicel p45).
+ Crownpak CR [755], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [766], matching ALANINE (Daicel p33).
+ Chiralpak WH [766], matching ALANINE (Daicel p33).
+ Chiralpak WM [766], matching ALANINE (Daicel p33).
+ Crownpak CR [845], matching L-3,4-DIHYDROXYPHENYLALANINE (Daicel p35).
+ Chiralcel OD [879], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralcel OB [888], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Crownpak CR [981], matching 1 PHENYLETHYLAMINE (Daicel p32).
+ Chiralcel OK [1011], matching METHYL MANDELATE (Daicel p44).
+ Chiralcel OB [1043], matching 2-ACETOXY-3-PHENYLPROPANE (Daicel p45).
+ Chiralcel OB [1052], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel CA-1 [1093], matching MANDELAMIDE (Daicel p44).
+ Chiralcel OB [1136], matching 1-PHENYLETHANOL (Daicel p42).
+ Chiralcel OB [1211], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OB [1271], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
+ Chiralpak WH [1329], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [1329], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [1363], matching PINDOLOL (Daicel p20).
+ Chiralcel OJ [1417], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1417], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OK [1417], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1417], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OC [1417], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [1417], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [1417], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [1442], matching FENOPROFEN (Daicel p27).
+ Chiralcel OD [1448], matching BENZOIN (Daicel p44).
+ Chiralcel OF [1448], matching BENZOIN (Daicel p44).
+ Chiralpak OP [1448], matching BENZOIN (Daicel p44).
+ Chiralcel OB [1465], matching METHYL-3-BENZOYLOXYPROPANOATE (Daicel p45).
+ Chiralpak OT [1476], matching 1-BROMO ETHYLBENZENE (Daicel p46).
+ Chiralcel OJ [1481], matching KETOPROFEN (Daicel p27).
+ Chiralcel OD [1502], matching BENZYL PHENYL CARBINOL (Daicel p46).
+ Chiralpak OT [1536], matching 2 ETHYL PHENYL BENZOATE (Daicel p46).
+ Chiralpak WH [1547], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1547], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OB [1580], matching ALPHA METHYL PHENETHYL BENZOATE (Daicel p45).
+ Chiralcel OB [1613], matching BUTYNYL 3 BENZOATE (Daicel p45).
+ Chiralcel OJ [1621], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OK [1621], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OD [1665], matching METOPROLOL (Daicel p20).
+ Chiralcel OD [1675], matching ALPRENOLOL (Daicel p20).
+ Chiralpak WH [1690], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1690], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1716], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OB [1752], matching 1 METHYL 3 PHENYL PROPYL BENZOATE (Daicel p45).
+ Chiralcel OD [1753], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [1753], matching PERISOXAL (Daicel p20).
+ Chiralcel OB [1783], matching 1,3 BUTANEDIOL DIBENZOATE (Daicel p45).
+ Chiralcel OD [1783], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [1783], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [1783], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).

+ Chiralcel CA-1 [1783], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p44}.
+ Chiralpak WH [1815], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1848], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [1872], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1929], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2001], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OB [2078], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2279], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule CITRULLINE....
+ Crownpak CR [353], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [353], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [386], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [438], matching ARGININE {Daicel p39}.
+ Chiralpak WE [438], matching ARGININE {Daicel p39}.
+ Chiralpak WM [438], matching ARGININE {Daicel p39}.
+ Crownpak CR [553], matching ASPARAGINE {Daicel p37}.
+ Crownpak CR [611], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [724], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [734], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [734], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [936], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [1077], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [1100], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1100], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Crownpak CR [1122], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1122], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1122], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [1152], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1152], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1152], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1152], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1271], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1271], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1271], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1276], matching N-AMIDE-L-PHENYLALANINE {Daicel p41}.
+ Chiralpak WH [1280], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [1280], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [1297], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralcel OB [1421], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralpak WH [1479], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OD [1481], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1506], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Crownpak CR [1562], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralcel CA-1 [1605], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [1622], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralpak WH [1638], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [1686], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [1704], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OD [1742], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OB [1943], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [1959], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralpak OT [2004], matching 1-BROMO ETHYLBENZENE {Daicel p46}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule DL-VALINE....
+ Crownpak CR [70], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [70], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [70], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [70], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [133], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralpak WE [146], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [146], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [199], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [206], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [219], matching ALANINE {Daicel p33}.
+ Chiralpak WH [219], matching ALANINE {Daicel p33}.
+ Chiralpak WM [219], matching ALANINE {Daicel p33}.
+ Chiralpak WM [223], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [223], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [223], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [223], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [344], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [359], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [359], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [368], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [368], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [368], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [514], matching THREONINE {Daicel p35}.
+ Chiralpak WH [514], matching THREONINE {Daicel p35}.
+ Crownpak CR [518], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [547], matching LYSINE {Daicel p38}.
+ Chiralpak WM [547], matching LYSINE {Daicel p38}.
+ Chiralpak WE [547], matching LYSINE {Daicel p38}.
+ Chiralpak WH [566], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [566], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [623], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Crownpak CR [634], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [634], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [634], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [634], matching ASPARTIC ACID {Daicel p37}.
+ Chiralcel OB [711], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralcel OD [739], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Crownpak CR [759], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [767], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [767], matching HISTIDINE {Daicel p38}.
+ Chiralcel OB [769], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel OK [855], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [865], matching ASPARAGINE {Daicel p37}.
+ Chiralcel OB [905], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel CA-1 [910], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [945], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [1020], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1051], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [1095], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1152], matching CITRULLINE {Daicel p37}.
+ Chiralcel OB [1291], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1310], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [1373], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralpak WH [1428], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1428], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OB [1429], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1429], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1429], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1429], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1429], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1429], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1429], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1444], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1444], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OD [1471], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [1473], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [1475], matching FENOPROFEN {Daicel p27}.
+ Chiralpak WH [1497], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1497], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1540], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1540], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1540], matching BENZOIN {Daicel p44}.
+ Chiralcel OB [1542], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [1631], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [1646], matching N-CBZ-ALANINE {Daicel p40}.

+ Chiralcel OD [1646], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OB [1682], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1735], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1735], matching PERISOXAL {Daicel p20}.
+ Chiralpak WH [1795], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1804], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1822], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1834], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1834], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1834], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1834], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OD [2008], matching METOPROLOL {Daicel p20}.
+ Chiralcel OB [2055], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [2061], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2208], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2245], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OD [2343], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2357], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule L-ISOLEUCINE....
+ Chiralpak WE [1], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [70], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [70], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [70], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [70], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [126], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [126], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [126], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [126], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [133], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [141], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [191], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [191], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [191], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [206], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [286], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [360], matching LYSINE {Daicel p38}.
+ Chiralpak WM [360], matching LYSINE {Daicel p38}.
+ Chiralpak WE [360], matching LYSINE {Daicel p38}.
+ Crownpak CR [361], matching ALANINE {Daicel p33}.
+ Chiralpak WH [361], matching ALANINE {Daicel p33}.
+ Chiralpak WM [361], matching ALANINE {Daicel p33}.
+ Crownpak CR [364], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [364], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [400], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralpak WH [507], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [507], matching MANDELIC ACID {Daicel p27}.
+ Chiralpak WH [535], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [575], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [601], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [636], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OB [652], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralcel OD [681], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [710], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel OK [796], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [807], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [836], matching L-GLUTAMINE {Daicel p36}.
+ Chiralcel CA-1 [852], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [874], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OR [886], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OR [962], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1036], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1048], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Chiralcel OB [1287], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1287], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1287], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1287], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1287], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1287], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1287], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1294], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralpak OT [1315], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [1333], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [1365], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1377], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1377], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1377], matching BENZOIN {Daicel p44}.
+ Chiralpak WH [1428], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1428], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OB [1452], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [1461], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [1489], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [1497], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1497], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OJ [1537], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1537], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1540], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1593], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1593], matching PERISOXAL {Daicel p20}.
+ Chiralcel OB [1616], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1618], matching N CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1662], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1680], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1692], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1692], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1692], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1692], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak WH [1729], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1729], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1866], matching METOPROLOL {Daicel p20}.
+ Chiralcel OB [1919], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2065], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2215], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2229], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OD [2339], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2418], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule L-ORNITHINE....
+ Crownpak CR [144], matching LYSINE {Daicel p38}.
+ Chiralpak WM [144], matching LYSINE {Daicel p38}.
+ Chiralpak WE [144], matching LYSINE {Daicel p38}.
+ Chiralpak WM [295], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [295], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [295], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [295], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [306], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [306], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [354], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [359], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [359], matching DL-VALINE {Daicel p33}.

+ Chiralpak WM [359], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [359], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [364], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [364], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [364], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [364], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [385], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [386], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WE [438], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [438], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [499], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [504], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [532], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [573], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [573], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [573], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [621], matching ALANINE {Daicel p33}.
+ Chiralpak WH [621], matching ALANINE {Daicel p33}.
+ Chiralpak WM [621], matching ALANINE {Daicel p33}.
+ Crownpak CR [734], matching CITRULLINE {Daicel p37}.
+ Chiralpak WH [753], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [753], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [756], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Crownpak CR [786], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [898], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralcel OD [926], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Crownpak CR [939], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralcel OB [956], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel OK [1042], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel CA-1 [1098], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [1167], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [1243], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1282], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [1284], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [1285], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak WH [1473], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1473], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OJ [1476], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1476], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OD [1506], matching PINDOLOL {Daicel p20}.
+ Chiralcel OB [1559], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1559], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1559], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1559], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1559], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1559], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1559], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralpak OT [1567], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [1584], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OD [1608], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1608], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1608], matching BENZOIN {Daicel p44}.
+ Chiralcel OJ [1623], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1676], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralpak WH [1692], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1692], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OB [1727], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1754], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1791], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1791], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1807], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [1817], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1858], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OB [1882], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1894], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1895], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1895], matching PERISOXAL {Daicel p20}.
+ Chiralcel OB [1924], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [1925], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1925], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1925], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1925], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak WH [1959], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1993], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2071], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2135], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [2144], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OB [2220], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule ARGININE....
+ Crownpak CR [438], matching CITRULLINE {Daicel p37}.
+ Crownpak CR [589], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [693], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [716], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Crownpak CR [854], matching ASPARAGINE {Daicel p37}.
+ Crownpak CR [1020], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1020], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1020], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OD [1203], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1203], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OD [1267], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OD [1333], matching GUAIFENESINE {Daicel p44}.
+ Chiralpak WH [1376], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OB [1563], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [1956], matching TRIADIMEFON {Daicel p22}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule L-GLUTAMIC ACID....
+ Crownpak CR [153], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [153], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [153], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [254], matching THREONINE {Daicel p35}.
+ Chiralpak WH [254], matching THREONINE {Daicel p35}.
+ Crownpak CR [254], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [254], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [317], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [354], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [354], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [403], matching SERINE {Daicel p35}.
+ Chiralpak WH [403], matching SERINE {Daicel p35}.
+ Crownpak CR [415], matching ASPARAGINE {Daicel p37}.
+ Crownpak CR [474], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [518], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [518], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [518], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [518], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [568], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [611], matching CITRULLINE {Daicel p37}.
+ Crownpak CR [636], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [636], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [636], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [636], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [639], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [639], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [643], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [643], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Crownpak CR [659], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [666], matching ALANINE {Daicel p33}.

```
+ Chiralpak WH [666], matching ALANINE (Daicel p33).
+ Chiralpak WM [666], matching ALANINE (Daicel p33).
+ Crownpak CR [709], matching L-3,4-DIHYDROXYPHENYLALANINE (Daicel p35).
+ Crownpak CR [713], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [788], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Chiralcel OD [818], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [818], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Chiralpak WH [935], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [935], matching MANDELIC ACID (Daicel p27).
+ Chiralpak WH [1031], matching LACTIC ACID (Daicel p27).
+ Chiralcel OB [1072], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Chiralcel OB [1134], matching 1-PHENYL-1,3-PROPANEDIOL (Daicel p44).
+ Chiralpak WH [1174], matching N-ACTEYL VALINE (Daicel p39).
+ Chiralcel CA-1 [1279], matching MANDELAMIDE (Daicel p44).
+ Crownpak CR [1359], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralpak WH [1441], matching N-ACTEYL-L-PHENYLALANINE (Daicel p39).
+ Chiralcel OJ [1758], matching TRIADIMEFON (Daicel p22).
+ Chiralpak WH [1842], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [1842], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [1904], matching PRACTOLOL (Daicel p20).
+ Chiralcel OD [2071], matching METOPROLOL (Daicel p20).

************************************************************
*** Testing for suggestions for column Crownpak CR ....
- Ordered column suggestions (column) for molecule ASPARAGINE....
+ Crownpak CR [173], matching L-GLUTAMINE (Daicel p36).
+ Crownpak CR [241], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WH [241], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [241], matching ASPARTIC ACID (Daicel p37).
+ Crownpak CR [268], matching THREONINE (Daicel p35).
+ Chiralpak WH [268], matching THREONINE (Daicel p35).
+ Crownpak CR [360], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [360], matching HISTIDINE (Daicel p38).
+ Crownpak CR [366], matching SERINE (Daicel p35).
+ Chiralpak WH [366], matching SERINE (Daicel p35).
+ Crownpak CR [415], matching L-GLUTAMIC ACID (Daicel p36).
+ Crownpak CR [553], matching CITRULLINE (Daicel p37).
+ Chiralpak WH [733], matching GLYCERIC ACID (Daicel p27).
+ Chiralpak WH [759], matching N-ACTEYL VALINE (Daicel p39).
+ Crownpak CR [854], matching ARGININE (Daicel p39).
+ Chiralpak WE [854], matching ARGININE (Daicel p39).
+ Chiralpak WM [854], matching ARGININE (Daicel p39).
+ Crownpak CR [865], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [865], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [865], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [865], matching DL-VALINE (Daicel p33).
+ Crownpak CR [983], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [983], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Crownpak CR [1017], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Chiralpak WH [1027], matching N-ACETYL-L-PHENYLALANINE (Daicel p39).
+ Chiralcel CA-1 [1085], matching MANDELAMIDE (Daicel p44).
+ Chiralpak WH [1139], matching LACTIC ACID (Daicel p27).
+ Chiralcel OB [1371], matching 1-PHENYL-1,3-PROPANEDIOL (Daicel p44).
+ Crownpak CR [1413], matching 2 AMINO BUTANE (Daicel p32).
+ Chiralpak WH [1428], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [1428], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralpak WH [1462], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [1462], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OB [1469], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Chiralcel OD [1735], matching GUALFENESINE (Daicel p44).
+ Chiralcel OD [1747], matching PRACTOLOL (Daicel p20).
+ Chiralcel OJ [1867], matching TRIADIMEFON (Daicel p22).

************************************************************
*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions (column) for molecule ASPARTIC ACID....
+ Crownpak CR [92], matching THREONINE (Daicel p35).
+ Chiralpak WH [92], matching THREONINE (Daicel p35).
+ Crownpak CR [153], matching L-GLUTAMIC ACID (Daicel p36).
+ Crownpak CR [162], matching SERINE (Daicel p35).
+ Chiralpak WH [162], matching SERINE (Daicel p35).
+ Crownpak CR [241], matching ASPARAGINE (Daicel p37).
+ Crownpak CR [386], matching L-GLUTAMINE (Daicel p36).
+ Crownpak CR [401], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [401], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [529], matching GLYCERIC ACID (Daicel p27).
+ Crownpak CR [592], matching ALANINE (Daicel p33).
+ Chiralpak WH [592], matching ALANINE (Daicel p33).
+ Chiralpak WM [592], matching ALANINE (Daicel p33).
+ Crownpak CR [628], matching DL-TYROSINE (Daicel p35).
+ Crownpak CR [634], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [634], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [634], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [634], matching DL-VALINE (Daicel p33).
+ Crownpak CR [781], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [781], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Crownpak CR [786], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Crownpak CR [873], matching L-3,4-DIHYDROXYPHENYLALANINE (Daicel p35).
+ Chiralpak WH [942], matching LACTIC ACID (Daicel p27).
+ Chiralpak WH [999], matching N-ACTEYL VALINE (Daicel p39).
+ Chiralcel OD [1032], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [1032], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Crownpak CR [1134], matching 2 AMINO BUTANE (Daicel p32).
+ Chiralpak WH [1265], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [1265], matching MANDELIC ACID (Daicel p27).
+ Chiralpak WH [1267], matching N-ACETYL-L-PHENYLALANINE (Daicel p39).
+ Chiralpak WH [1668], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [1668], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OJ [1848], matching TRIADIMEFON (Daicel p22).
+ Chiralcel OD [2051], matching PRACTOLOL (Daicel p20).
+ Chiralcel OD [2209], matching METOPROLOL (Daicel p20).

************************************************************
*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions (column) for molecule THREONINE....
+ Crownpak CR [73], matching SERINE (Daicel p35).
+ Chiralpak WH [73], matching SERINE (Daicel p35).
+ Crownpak CR [92], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WH [92], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [92], matching ASPARTIC ACID (Daicel p37).
+ Crownpak CR [254], matching L-GLUTAMIC ACID (Daicel p36).
+ Crownpak CR [268], matching ASPARAGINE (Daicel p37).
+ Crownpak CR [364], matching L-GLUTAMINE (Daicel p36).
+ Crownpak CR [434], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [434], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [439], matching GLYCERIC ACID (Daicel p27).
+ Crownpak CR [471], matching ALANINE (Daicel p33).
+ Chiralpak WH [471], matching ALANINE (Daicel p33).
+ Chiralpak WM [471], matching ALANINE (Daicel p33).
+ Crownpak CR [514], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [514], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [514], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [514], matching DL-VALINE (Daicel p33).
+ Crownpak CR [666], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Crownpak CR [707], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [707], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [838], matching LACTIC ACID (Daicel p27).
+ Crownpak CR [962], matching L-3,4-DIHYDROXYPHENYLALANINE (Daicel p35).
+ Crownpak CR [1019], matching 2 AMINO BUTANE (Daicel p32).
+ Chiralpak WH [1027], matching N-ACTEYL VALINE (Daicel p39).
+ Chiralcel OB [1068], matching 1-PHENYL-1,3-PROPANEDIOL (Daicel p44).
+ Chiralcel OB [1140], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Chiralpak WH [1200], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [1200], matching MANDELIC ACID (Daicel p27).
+ Chiralpak WH [1294], matching N-ACETYL-L-PHENYLALANINE (Daicel p39).
+ Chiralcel OD [1406], matching GUALFENESINE (Daicel p44).
+ Chiralcel CA-1 [1437], matching MANDELAMIDE (Daicel p44).
+ Chiralpak WH [1695], matching N-CBZ-ASPARAGINE (Daicel p40).
```

+ Chiralcel OD [1695], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1809], matching PINDOLOL {Daicel p20}.
+ Chiralcel OJ [1886], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OD [2120], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [2318], matching PRACTOLOL {Daicel p20}.

****

*** Testing for suggestions for column Crownpak CR ....
- Ordered column suggestions [column] for molecule SERINE....
+ Crownpak CR [73], matching THREONINE {Daicel p35}.
+ Chiralpak WH [73], matching THREONINE {Daicel p35}.
+ Crownpak CR [162], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [162], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [162], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [344], matching GLYCERIC ACID {Daicel p27}.
+ Crownpak CR [366], matching ASPARAGINE {Daicel p37}.
+ Crownpak CR [403], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [500], matching ALANINE {Daicel p33}.
+ Chiralpak WH [500], matching ALANINE {Daicel p33}.
+ Chiralpak WM [500], matching ALANINE {Daicel p33}.
+ Crownpak CR [577], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [600], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [600], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [723], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [861], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [861], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [1045], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [1045], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [1068], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [1111], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Chiralpak WH [1125], matching N-ACTEYL VALINE {Daicel p39}.
+ Crownpak CR [1141], matching 2 AMINO BUTANE {Daicel p32}.
+ Chiralcel OB [1223], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel OD [1334], matching GUAIFENESINE {Daicel p44}.
+ Chiralcel CA 1 [1347], matching MANDELAMIDE {Daicel p44}.
+ Chiralpak WH [1393], matching N-ACETYL-L PHENYLALANINE {Daicel p39}.
+ Chiralpak WH [1793], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1793], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1942], matching PINDOLOL {Daicel p20}.
+ Chiralcel OJ [2086], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OD [2189], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OD [2202], matching METOPROLOL {Daicel p20}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule ALANINE....
+ Crownpak CR [219], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [219], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [219], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [219], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [344], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [352], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [361], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [361], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [361], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [361], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [365], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [365], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [443], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [471], matching THREONINE {Daicel p35}.
+ Chiralpak WH [471], matching THREONINE {Daicel p35}.
+ Chiralpak WM [480], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [480], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [480], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [480], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [494], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [500], matching SERINE {Daicel p35}.
+ Chiralpak WH [500], matching SERINE {Daicel p35}.
+ Crownpak CR [587], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [587], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [587], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [592], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [592], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [592], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [621], matching L ORNITHINE {Daicel p38}.
+ Chiralpak WM [621], matching L ORNITHINE {Daicel p38}.
+ Crownpak CR [642], matching 2 AMINO BUTANE {Daicel p32}.
+ Crownpak CR [666], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OB [673], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Crownpak CR [766], matching LYSINE {Daicel p38}.
+ Chiralpak WM [766], matching LYSINE {Daicel p38}.
+ Chiralpak WE [766], matching LYSINE {Daicel p38}.
+ Crownpak CR [775], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Crownpak CR [841], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [844], matching GLYCERIC ACID {Daicel p27}.
+ Crownpak CR [857], matching L-GLUTAMINE {Daicel p36}.
+ Chiralcel OD [863], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Crownpak CR [891], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [891], matching HISTIDINE {Daicel p38}.
+ Chiralcel OJ [1006], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1006], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1063], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [1068], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [1078], matching METHYL 3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [1095], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1103], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [1144], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1219], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [1249], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Crownpak CR [1271], matching CITRULLINE {Daicel p37}.
+ Chiralpak WH [1428], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1428], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OJ [1438], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1438], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1438], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1438], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1438], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1438], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1438], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralpak OT [1497], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [1497], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [1537], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1607], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1607], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1607], matching BENZOIN {Daicel p44}.
+ Chiralcel OB [1618], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralpak WH [1646], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1646], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1654], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [1661], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1789], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1789], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OB [1801], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [1807], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OB [1831], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [1836], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1836], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1836], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1836], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OD [1876], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1876], matching PERISOXAL {Daicel p20}.

+ Chiralcel OD [1957], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1975], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OB [1977], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [2014], matching N-CBZ PHENYLALANINE {Daicel p40}.
+ Chiralcel OB [2127], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [2161], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [2609], matching CARTEOLOL {Daicel p20}.

****

*** Testing for suggestions for column Crownpak CR ....
= Ordered column suggestions [column] for molecule 2 AMINO BUTANE....
+ Crownpak CR [367], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Crownpak CR [394], matching 2-AMINO-3 METHYLBUTANE {Daicel p32}.
+ Crownpak CR [642], matching ALANINE {Daicel p33}.
+ Chiralpak WH [642], matching ALANINE {Daicel p33}.
+ Chiralpak WM [642], matching ALANINE {Daicel p33}.
+ Chiralcel OJ [782], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralpak OT [872], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Crownpak CR [1019], matching THREONINE {Daicel p35}.
+ Chiralpak WH [1019], matching THREONINE {Daicel p35}.
+ Chiralpak WH [1023], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [1134], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [1134], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [1134], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [1141], matching SERINE {Daicel p35}.
+ Chiralpak WH [1141], matching SERINE {Daicel p35}.
+ Crownpak CR [1413], matching ASPARAGINE {Daicel p37}.
+ Chiralpak WH [1523], matching GLYCERIC ACID {Daicel p27}.
+ Chiralpak OT [1525], matching ETHYL 2,3 DIPHENYL PROPANOATE {Daicel p46}.
+ Chiralcel OJ [1562], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralcel OB [1563], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1-PHENYLETHANOL....
+ Chiralcel OB [63], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [183], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OD [206], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OK [289], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [344], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralcel OB [419], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [419], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [419], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [419], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [419], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [419], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [419], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Crownpak CR [463], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [471], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel OB [500], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralcel OB [558], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OJ [562], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OD [587], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [599], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OD [618], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [618], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralpak WH [626], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [626], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [647], matching 1.3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [656], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [656], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [656], matching BENZOIN {Daicel p44}.
+ Crownpak CR [733], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [765], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [775], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [775], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [775], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [775], matching L-METHIONINE {Daicel p36}.
+ Chiralcel OD [840], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OC [840], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [840], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [840], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OJ [883], matching FENOPROFEN {Daicel p27}.
+ Crownpak CR [889], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OJ [895], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [895], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Crownpak CR [921], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [921], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [921], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [921], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak OT [930], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Crownpak CR [936], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [936], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [936], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [948], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralpak WE [973], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [973], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [980], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [980], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [980], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [980], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1058], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Chiralcel OB [1085], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Crownpak CR [1103], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1103], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1103], matching ALANINE {Daicel p33}.
+ Chiralcel OJ [1108], matching KETOPROFEN {Daicel p27}.
+ Crownpak CR [1119], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1119], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1119], matching LYSINE {Daicel p38}.
+ Chiralpak WH [1132], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [1167], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1167], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OB [1225], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1348], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1348], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OB [1370], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1393], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1393], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1414], matching ALPRENOLOL {Daicel p20}.
+ Crownpak CR [1419], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [1422], matching PROPRANOLOL {Daicel p20}.
+ Chiralpak WH [1567], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1567], matching N-CBZ-VALINE {Daicel p40}.
+ Crownpak CR [1593], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1593], matching HISTIDINE {Daicel p38}.
+ Chiralcel OD [1651], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OB [1653], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OD [1669], matching METOPROLOL {Daicel p20}.
+ Chiralpak WH [1709], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1709], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OJ [1838], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1838], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1853], matching CARBINOXAMINE {Daicel p21}.
+ Chiralpak WH [1934], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Crownpak CR [1943], matching CITRULLINE {Daicel p37}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1-PHENYL-PROPAN-1-OL....
+ Chiralcel OB [63], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OK [206], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [245], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.

+ Chiralcel OD [269], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [394], matching 1-PHENYL-1,3 PROPANEDIOL {Daicel p44}.
+ Crownpak CR [430], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Crownpak CR [433], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [495], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [495], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [495], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [495], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [495], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [495], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [495], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [511], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [516], matching 1 PHENYL 1,2 ETHANEDIOL {Daicel p44}.
+ Chiralcel OD [581], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [581], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [581], matching BENZOIN {Daicel p44}.
+ Chiralcel OJ [624], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OB [633], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralpak OT [634], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralpak WH [642], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [642], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [662], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OD [680], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [680], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OB [710], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel CA-1 [789], matching MANDELAMIDE {Daicel p44}.
+ Crownpak CR [825], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [857], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [868], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [868], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [868], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [868], matching L-METHIONINE {Daicel p36}.
+ Chiralcel OD [916], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [916], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [916], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [916], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OJ [957], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [957], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OJ [959], matching FENOPROFEN {Daicel p27}.
+ Crownpak CR [965], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralpak OT [992], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Crownpak CR [997], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [997], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [997], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [997], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [1012], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1012], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1012], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [1024], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [1042], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Chiralpak WE [1048], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1048], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1055], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1055], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1055], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1055], matching DL-VALINE {Daicel p33}.
+ Chiralcel OB [1161], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Crownpak CR [1179], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1179], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1179], matching ALANINE {Daicel p33}.
+ Chiralcel OJ [1184], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [1208], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [1211], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1211], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1211], matching LYSINE {Daicel p38}.
+ Crownpak CR [1243], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1243], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OB [1301], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1317], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1317], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1338], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1346], matching PROPRANOLOL {Daicel p20}.
+ Chiralpak WH [1424], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1424], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OB [1432], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1491], matching N CBZ VALINE {Daicel p40}.
+ Chiralcel OD [1491], matching N CBZ VALINE {Daicel p40}.
+ Crownpak CR [1511], matching ERYTHRO 2 AMINO 1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [1590], matching N,O-DIBENZOYL2 AMINOBUTANOL {Daicel p46}.
+ Chiralcel OD [1593], matching METOPROLOL {Daicel p20}.
+ Crownpak CR [1609], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1609], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1633], matching N-CBZ-NOR LEUCINE {Daicel p40}.
+ Chiralcel OD [1633], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1727], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OJ [1762], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1762], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1777], matching CARBINOXAMINE {Daicel p21}.
+ Chiralpak WH [1859], matching N-CBZ PHENYLALANINE {Daicel p40}.
+ Crownpak CR [1959], matching CITRULLINE {Daicel p37}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 2 ACETOXY-3-PHENYLPROPANE....
+ Chiralcel OD [298], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [317], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [395], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OJ [432], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OB [464], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [507], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Crownpak CR [516], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Chiralcel OB [558], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [615], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [615], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Crownpak CR [617], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [617], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [617], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OB [633], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [647], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Crownpak CR [651], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralpak OT [662], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [667], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [668], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [668], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [668], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [668], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [668], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [668], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [668], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [707], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [707], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralpak WM [728], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [728], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [728], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [728], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [835], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [835], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [835], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [835], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [835], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [835], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [838], matching PHENYLGLYCINE {Daicel p34}.

+ Chiralcel OD [845], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OD [855], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [855], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [855], matching BENZOIN {Daicel p44}.
+ Chiralcel OK [857], matching METHYL MANDELATE {Daicel p44}.
+ Chiralpak WH [866], matching N ACETYL-L PHENYLALANINE {Daicel p39}.
+ Chiralcel OJ [894], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [900], matching 2-AMINO-3 PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OB [980], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Crownpak CR [1020], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [1042], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Crownpak CR [1043], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1043], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1043], matching LYSINE {Daicel p38}.
+ Chiralcel OB [1055], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [1062], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1079], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1079], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1079], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1079], matching DL-VALINE {Daicel p33}.
+ Chiralcel OD [1089], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1089], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1089], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1089], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak WH [1098], matching N-AMIDE-L-PHENYLALANINE {Daicel p41}.
+ Chiralpak WH [1134], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1134], matching N-CBZ-ALANINE {Daicel p40}.
+ Crownpak CR [1197], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1197], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1197], matching ALANINE {Daicel p33}.
+ Chiralcel OB [1208], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1235], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Crownpak CR [1284], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1284], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OD [1306], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralpak WH [1353], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1353], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OB [1357], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [1429], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1429], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralpak WH [1626], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1647], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1647], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1663], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1671], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OB [1762], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [1842], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OD [1927], matching METOPROLOL {Daicel p20}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 2-PHENYL-PROPAN-1-OL....
+ Chiralcel OD [133], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [183], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [259], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [339], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralcel OJ [343], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OB [366], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralpak OT [387], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OB [395], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [443], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [466], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [466], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OK [482], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [524], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OJ [602], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [602], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [602], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [602], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [602], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [602], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [602], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [625], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [645], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OJ [677], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [677], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Crownpak CR [678], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [681], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralpak WM [689], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [689], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [689], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [689], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [739], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralpak WH [757], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [757], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [769], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralpak OT [794], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [825], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OD [839], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [839], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [839], matching BENZOIN {Daicel p44}.
+ Chiralcel CA-1 [859], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [949], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralpak WH [974], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OD [1022], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1022], matching 9 (2 TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1022], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1022], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Crownpak CR [1052], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1052], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1052], matching LYSINE {Daicel p38}.
+ Chiralcel OB [1089], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [1150], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Crownpak CR [1172], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Crownpak CR [1272], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1272], matching HISTIDINE {Daicel p38}.
+ Chiralcel OD [1443], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OB [1463], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OD [1581], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1581], matching PERISOXAL {Daicel p20}.
+ Chiralcel OJ [1593], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1593], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1596], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1604], matching PROPRANOLOL {Daicel p20}.
! Crownpak CR [1622], matching CITRULLINE {Daicel p37}.
+ Chiralcel OD [1680], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OD [1878], matching METOPROLOL {Daicel p20}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1-PHENYL-1,2-ETHANEDIOL....
+ Chiralcel OB [73], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralpak WH [133], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [133], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel CA-1 [234], matching MANDELAMIDE {Daicel p44}.
+ Crownpak CR [288], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OK [343], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [344], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.

+ Crownpak CR [476], matching PHENYLGLYCINE [Daicel p34].
+ Chiralpak WM [487], matching L METHIONINE [Daicel p36].
+ Crownpak CR [487], matching L-METHIONINE [Daicel p36].
+ Chiralpak WH [487], matching L-METHIONINE [Daicel p36].
+ Chiralpak WE [487], matching L-METHIONINE [Daicel p36].
+ Chiralcel OB [500], matching 1 PHENYLETHANOL [Daicel p42].
+ Crownpak CR [507], matching 2-AMINO-1-HEXANOL [Daicel p32].
+ Chiralcel OB [516], matching 1-PHENYL-PROPAN-1-OL [Daicel p42].
+ Chiralcel OD [541], matching GUALFENESINE [Daicel p44].
+ Crownpak CR [566], matching 2-AMINO-1-PENTANOL [Daicel p32].
+ Chiralcel OB [625], matching 2-PHENYL-PROPAN-1-OL [Daicel p42].
+ Crownpak CR [652], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WH [652], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WM [652], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WE [652], matching L-ISOLEUCINE [Daicel p34].
+ Crownpak CR [711], matching DL-VALINE [Daicel p33].
+ Chiralpak WH [711], matching DL-VALINE [Daicel p33].
+ Chiralpak WM [711], matching DL-VALINE [Daicel p33].
+ Chiralpak WE [711], matching DL-VALINE [Daicel p33].
+ Chiralcel OD [757], matching 2-PHENYLPROPIONIC ACID [Daicel p27].
+ Chiralpak WH [793], matching N ACETYL-L-PHENYLALANINE [Daicel p39].
+ Chiralpak WH [806], matching N-ACTEYL VALINE [Daicel p39].
+ Crownpak CR [881], matching 1 PHENYLETHYLAMINE [Daicel p32].
+ Crownpak CR [888], matching LYSINE [Daicel p38].
+ Chiralpak WM [888], matching LYSINE [Daicel p38].
+ Chiralpak WE [888], matching LYSINE [Daicel p38].
+ Crownpak CR [898], matching L-ORNITHINE [Daicel p38].
+ Chiralpak WM [898], matching L-ORNITHINE [Daicel p38].
+ Chiralpak OT [1011], matching 1-BROMO ETHYLBENZENE [Daicel p46].
+ Crownpak CR [1070], matching HISTIDINE [Daicel p38].
+ Chiralpak WH [1070], matching HISTIDINE [Daicel p38].
+ Crownpak CR [1080], matching L-GLUTAMIC ACID [Daicel p36].
+ Chiralcel OD [1090], matching 2PHENOXYPROPIONIC ACID [Daicel p29].
+ Chiralcel OF [1090], matching 2PHENOXYPROPIONIC ACID [Daicel p26].
+ Crownpak CR [1103], matching SERINE [Daicel p35].
+ Chiralpak WH [1103], matching SERINE [Daicel p35].
+ Crownpak CR [1271], matching THREONINE [Daicel p35].
+ Chiralpak WH [1271], matching THREONINE [Daicel p35].
+ Crownpak CR [1421], matching CITRULLINE [Daicel p37].
+ Chiralcel OD [1447], matching PINDOLOL [Daicel p20].
+ Crownpak CR [1500], matching ASPARAGINE [Daicel p37].
+ Crownpak CR [1680], matching L-GLUTAMINE [Daicel p36].
+ Chiralpak WH [1745], matching N-CBZ-ASPARAGINE [Daicel p40].
+ Chiralcel OD [1745], matching N-CBZ-ASPARAGINE [Daicel p40].
+ Crownpak CR [1983], matching D-(+)-TRYPTOPHAN [Daicel p38].
+ Chiralpak WH [1983], matching D-(+)-TRYPTOPHAN [Daicel p38].

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1-PHENYL-1,3-PROPANEDIOL....
+ Chiralcel OB [73], matching 1-PHENYL-1,2-ETHANEDIOL [Daicel p44].
+ Chiralcel OK [168], matching METHYL MANDELATE [Daicel p44].
+ Chiralpak WH [200], matching MANDELIC ACID [Daicel p27].
+ Chiralcel OD [200], matching MANDELIC ACID [Daicel p27].
+ Crownpak CR [269], matching 2-AMINO-1-PHENYLETHANOL [Daicel p32].
+ Chiralcel CA-1 [334], matching MANDELAMIDE [Daicel p44].
+ Chiralcel OB [393], matching 1-PHENYL-PROPAN-1-OL [Daicel p42].
+ Crownpak CR [439], matching 2-AMINO-2-PHENYLETHANOL [Daicel p32].
+ Chiralcel OB [471], matching 1-PHENYLETHANOL [Daicel p42].
+ Crownpak CR [566], matching PHENYLGLYCINE [Daicel p34].
+ Chiralpak WM [577], matching L-METHIONINE [Daicel p36].
+ Crownpak CR [577], matching L-METHIONINE [Daicel p36].
+ Chiralpak WH [577], matching L-METHIONINE [Daicel p36].
+ Chiralpak WE [577], matching L-METHIONINE [Daicel p36].
+ Crownpak CR [592], matching 2-AMINO-1-HEXANOL [Daicel p32].
+ Crownpak CR [650], matching 2 AMINO 1 PENTANOL [Daicel p42].
+ Chiralcel OD [709], matching GUALFENESINE [Daicel p44].
+ Crownpak CR [710], matching L ISOLEUCINE [Daicel p34].
+ Chiralpak WH [710], matching L ISOLEUCINE [Daicel p34].
+ Chiralpak WM [710], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WE [710], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WH [734], matching N-ACTEYL VALINE [Daicel p39].
+ Chiralpak WH [742], matching N-ACETYL-L-PHENYLALANINE [Daicel p39].
+ Crownpak CR [769], matching DL-VALINE [Daicel p33].
+ Chiralpak WH [769], matching DL-VALINE [Daicel p33].
+ Chiralpak WM [769], matching DL-VALINE [Daicel p33].
+ Chiralpak WE [769], matching DL-VALINE [Daicel p33].
+ Chiralcel OJ [792], matching METHYL2-PHENYLPROPIONATE [Daicel p28].
+ Crownpak CR [838], matching 1-PHENYLETHYLAMINE [Daicel p32].
+ Chiralcel OB [892], matching N-(PHENYLETHYL)ACETAMIDE [Daicel p46].
+ Crownpak CR [956], matching L-ORNITHINE [Daicel p38].
+ Chiralpak WM [956], matching L-ORNITHINE [Daicel p38].
+ Crownpak CR [1103], matching THREONINE [Daicel p35].
+ Chiralpak WH [1103], matching THREONINE [Daicel p35].
+ Chiralcel OJ [1125], matching METHYL 2 PHENOXYPROPIONATE [Daicel p29].
+ Chiralcel OK [1125], matching METHYL 2 PHENOXYPROPIONATE [Daicel p29].
+ Crownpak CR [1138], matching L-GLUTAMIC ACID [Daicel p36].
+ Crownpak CR [1156], matching HISTIDINE [Daicel p38].
+ Chiralpak WH [1156], matching HISTIDINE [Daicel p38].
+ Crownpak CR [1271], matching SERINE [Daicel p35].
+ Chiralpak WH [1271], matching SERINE [Daicel p35].
+ Chiralcel OD [1280], matching PINDOLOL [Daicel p20].
+ Crownpak CR [1449], matching ASPARAGINE [Daicel p37].
+ Crownpak CR [1506], matching CITRULLINE [Daicel p37].
+ Crownpak CR [1513], matching L-GLUTAMINE [Daicel p36].
+ Chiralpak WH [1694], matching N-CBZ-ASPARAGINE [Daicel p40].
+ Chiralcel OD [1694], matching N-CBZ-ASPARAGINE [Daicel p40].
+ Crownpak CR [1816], matching D-(+)-TRYPTOPHAN [Daicel p38].
+ Chiralpak WH [1816], matching D-(+)-TRYPTOPHAN [Daicel p38].

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule
N,O-DIBENZOYL2-AMINOBUTANOL....
+ Chiralcel OB [651], matching 1 METHYL 3 PHENYL PROPYL BENZOATE [Daicel p45].
+ Chiralcel OB [1102], matching N-(PHENYLETHYL)ACETAMIDE [Daicel p46].
+ Chiralcel OJ [1147], matching CYCLOPENTOLATE [Daicel p21].
+ Chiralcel OD [1147], matching CYCLOPENTOLATE [Daicel p21].
+ Chiralcel OD [1157], matching 2 BUTYL PHENYL CARBAMATE [Daicel p46].
+ Chiralcel OD [1230], matching CARBINOXAMINE [Daicel p21].
+ Chiralcel OJ [1350], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OB [1350], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OK [1350], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OB [1350], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OC [1350], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OJ [1350], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OD [1350], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Crownpak CR [1409], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL [Daicel p32].
+ Chiralcel OB [1443], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE [Daicel p29].
+ Chiralcel OB [1463], matching 2-PHENYL-PROPAN-1-OL [Daicel p42].
+ Chiralcel OB [1590], matching 1-PHENYL-PROPAN-1-OL [Daicel p42].
+ Chiralpak WH [1625], matching 2-AMINO-3-PHENYLPROPIONAMIDE [Daicel p41].
+ Chiralcel OB [1666], matching 1-PHENYLETHANOL [Daicel p42].
+ Chiralcel OD [1743], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE [Daicel p43].
+ Chiralcel OG [1743], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE [Daicel p43].
+ Chiralpak OT [1743], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE [Daicel p43].
+ Chiralcel CA-1 [1743], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE [Daicel p43].
+ Chiralcel OD [1743], matching BENZOIN [Daicel p44].
+ Chiralcel OF [1743], matching BENZOIN [Daicel p44].
+ Chiralpak OP [1743], matching BENZOIN [Daicel p44].
+ Chiralpak WH [1750], matching N ACTEYL VALINE [Daicel p39].
+ Chiralcel OD [1813], matching BENZYL PHENYL CARBINOL [Daicel p46].
+ Chiralcel OK [1844], matching METHYL MANDELATE [Daicel p44].

+ Chiralpak WH [1865], matching WARFARINE {Daicel p39}.
+ Crownpak CR [2004], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [2037], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [2084], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OD [2150], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [2153], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [2191], matching PROPRANOLOL {Daicel p20}.
+ Crownpak CR [2226], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralcel OD [2233], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [2233], matching PERISOXAL {Daicel p20}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule
ALPHA METHYL PHENETHYL BENZOATE....
+ Chiralcel OB [141], matching 1-METHYL-3-PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralpak OT [156], matching 2-ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OB [507], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [512], matching PENTYNYL-3-BENZOATE {Daicel p45}.
+ Chiralcel OB [513], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [520], matching 1,3-BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [666], matching 1,4-PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [667], matching BUTYNYL-3-BENZOATE {Daicel p45}.
+ Chiralcel OD [790], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [798], matching 3-PHENYL-BUTYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [801], matching METHYL-2-PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [801], matching METHYL-2-PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [816], matching 1,5-HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [867], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OJ [944], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OB [949], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1046], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p45}.
+ Chiralcel OB [1085], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [1101], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1101], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OJ [1115], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OJ [1158], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [1161], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1175], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1175], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1175], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OB [1180], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [1196], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1196], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1196], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1196], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1196], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1196], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1196], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralpak WH [1228], matching WARFARINE {Daicel p39}.
+ Crownpak CR [1330], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [1355], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1355], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1355], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1355], matching L-METHIONINE {Daicel p36}.
+ Chiralcel OJ [1358], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [1378], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralpak WE [1397], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1397], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1400], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1400], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1400], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1400], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [1542], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1542], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1542], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1542], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1598], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1598], matching N-CBZ-ALANINE {Daicel p40}.
+ Crownpak CR [1605], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1605], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1605], matching LYSINE {Daicel p38}.
+ Chiralcel OD [1616], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1616], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1616], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1616], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Crownpak CR [1661], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1661], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1661], matching ALANINE {Daicel p33}.
+ Crownpak CR [1747], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1747], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WH [1817], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1817], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [1959], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1959], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralpak WH [2184], matching N-CBZ-PHENYLALANINE {Daicel p40}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1,3-DIACETOXY-BUTANE....
+ Chiralcel OB [317], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [349], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [443], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [507], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OJ [548], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OB [606], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [647], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [675], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [675], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Crownpak CR [720], matching LYSINE {Daicel p38}.
+ Chiralpak WM [720], matching LYSINE {Daicel p38}.
+ Chiralpak WE [720], matching LYSINE {Daicel p38}.
+ Chiralcel OB [723], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [728], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [728], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [740], matching ALANINE {Daicel p33}.
+ Chiralpak WH [740], matching ALANINE {Daicel p33}.
+ Chiralpak WM [740], matching ALANINE {Daicel p33}.
+ Chiralpak WM [757], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [757], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [757], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [757], matching L-METHIONINE {Daicel p36}.
+ Chiralcel OJ [770], matching ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE {Daicel
p29}.
+ Chiralpak OT [787], matching 2-ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralpak OT [802], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OJ [804], matching METHYL 2-PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [804], matching METHYL 2-PHENOXYPROPIONATE {Daicel p29}.
+ Crownpak CR [830], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p45}.
+ Chiralcel OB [859], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Crownpak CR [861], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [874], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [874], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [874], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [874], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [875], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [875], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [892], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [892], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [892], matching DL-VALINE {Daicel p33}.

+ Chiralpak WE [892], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [933], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [933], matching D PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [933], matching D PHENYLALANINE {Daicel p34}.
+ Chiralcel OK [947], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [951], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [967], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [967], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [967], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [967], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [967], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [967], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [967], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Crownpak CR [978], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralcel OJ [1001], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [1010], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1011], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OJ [1041], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [1042], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1088], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1088], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1107], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1107], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1107], matching BENZOIN {Daicel p44}.
+ Crownpak CR [1110], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [1120], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OB [1158], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [1161], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OD [1206], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralpak WH [1240], matching N CBZ VALINE {Daicel p40}.
+ Chiralcel OD [1240], matching N CBZ VALINE {Daicel p40}.
+ Chiralcel OB [1307], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [1366], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1366], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1366], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1366], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak WH [1382], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1382], matching N-CBZ NOR LEUCINE {Daicel p40}.
+ Chiralcel OB [1462], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralpak WH [1493], matching N-AMIDE-L-PHENYLALANINE {Daicel p41}.
+ Chiralcel OB [1543], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralpak WH [1607], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1900], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [1925], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1966], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [2001], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [2001], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [2297], matching CARTEOLOL {Daicel p20}.

****
*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule METHYL-3-BENZOYLOXYPROPANOATE....
+ Chiralpak OT [307], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OB [323], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [396], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [396], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OJ [458], matching ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE {Daicel p29}.
+ Chiralcel OB [513], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OB [562], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [653], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [658], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [661], matching 1,3-DIACETOXY BUTANE {Daicel p45}.
+ Chiralcel OB [727], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OB [804], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [912], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OB [954], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [972], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY] PROPIONATE {Daicel p29}.
+ Chiralcel OJ [977], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralcel OD [983], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1020], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [1078], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1078], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1078], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1158], matching WARFARINE {Daicel p39}.
+ Crownpak CR [1286], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p45}.
+ Crownpak CR [1297], matching DL VALINE {Daicel p33}.
+ Chiralpak WH [1297], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1297], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1297], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1314], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1314], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [1439], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1439], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1439], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1439], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1442], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1442], matching DL LEUCINE {Daicel p33}.
+ Chiralpak WM [1446], matching L METHIONINE {Daicel p36}.
+ Crownpak CR [1446], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1446], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1446], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1458], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1458], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1458], matching LYSINE {Daicel p38}.
+ Crownpak CR [1469], matching 2-AMINO 1-PENTANOL {Daicel p32}.
+ Chiralcel OJ [1518], matching KETOPROFEN {Daicel p27}.
+ Crownpak CR [1520], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WH [1529], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [1611], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OJ [1618], matching FENOPROFEN {Daicel p27}.
+ Crownpak CR [1664], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1664], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1664], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1704], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1704], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralpak WH [1923], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1923], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [2065], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [2065], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralpak WH [2291], matching N-CBZ-PHENYLALANINE {Daicel p40}.

****
*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1-(1-HYDROXYETHYL)NAPHTHALENE....
+ Chiralcel OB [419], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [421], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [421], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [421], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OJ [465], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [495], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OJ [567], matching KETOPROFEN {Daicel p27}.

+ Chiralcel OB [602], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OD [625], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralcel OB [668], matching 2 ACETOXY-3-PHENYLPROPANE (Daicel p45).
+ Chiralcel OD [703], matching BENZOIN (Daicel p44).
+ Chiralcel OF [703], matching BENZOIN (Daicel p44).
+ Chiralpak OP [703], matching BENZOIN (Daicel p44).
+ Chiralcel OD [709], matching BENZYL PHENYL CARBINOL (Daicel p46).
+ Chiralcel OK [719], matching METHYL MANDELATE (Daicel p44).
+ Chiralcel OJ [818], matching TIAPROFENIC ACID ME ESTER (Daicel p28).
+ Crownpak CR [882], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralpak WH [904], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [904], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OB [967], matching 1,3-DIACETOXY-BUTANE (Daicel p45).
+ Chiralcel OD [970], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OJ [980], matching METHYL2-PHENYLPROPIONATE (Daicel p28).
+ Chiralcel OD [981], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [981], matching PERISOXAL (Daicel p20).
+ Chiralcel OD [1003], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OD [1006], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [1006], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Chiralcel OB [1018], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
+ Crownpak CR [1047], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [1047], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [1047], matching D-PHENYLALANINE (Daicel p34).
+ Chiralcel OB [1052], matching 1 METHYL 3 PHENYL PROPYL BENZOATE (Daicel p45).
+ Chiralpak WH [1123], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1123], matching N-CBZ-VALINE (Daicel p40).
+ Crownpak CR [1168], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Chiralcel OB [1196], matching ALPHA METHYL PHENETHYL BENZOATE (Daicel p45).
+ Crownpak CR [1200], matching PHENYLGLYCINE (Daicel p34).
+ Chiralcel OD [1210], matching METOPROLOL (Daicel p20).
+ Chiralcel OB [1210], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE (Daicel p29).
+ Chiralpak WM [1216], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [1216], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1216], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [1216], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1243], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Crownpak CR [1251], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID (Daicel p35).
+ Chiralpak WH [1265], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1265], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OJ [1282], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OK [1282], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Crownpak CR [1290], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Crownpak CR [1314], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [1314], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WE [1318], matching DL-LEUCINE (Daicel p33).
+ Chiralpak WM [1318], matching DL-LEUCINE (Daicel p33).
+ Crownpak CR [1322], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WH [1322], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [1322], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [1322], matching L-ISOLEUCINE (Daicel p34).
+ Chiralcel OB [1336], matching N,O-DIBENZOYL2-AMINOBUTANOL (Daicel p46).
+ Chiralpak OT [1349], matching 2 ETHYL PHENYL BENZOATE (Daicel p46).
+ Crownpak CR [1417], matching LYSINE (Daicel p38).
+ Chiralpak WM [1417], matching LYSINE (Daicel p38).
+ Chiralpak WE [1417], matching LYSINE (Daicel p38).
+ Crownpak CR [1432], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Crownpak CR [1464], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [1464], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [1464], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [1464], matching DL-VALINE (Daicel p33).
+ Crownpak CR [1473], matching ALANINE (Daicel p33).
+ Chiralpak WH [1473], matching ALANINE (Daicel p33).
+ Chiralpak WM [1473], matching ALANINE (Daicel p33).
+ Chiralpak WH [1490], matching N-CBZ-PHENYLALANINE (Daicel p40).
+ Crownpak CR [1559], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [1559], matching L-ORNITHINE (Daicel p38).
+ Chiralcel OJ [1767], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OD [1767], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OB [1788], matching 3 PHENYL BUTYL BENZOATE (Daicel p45).
+ Crownpak CR [1854], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL (Daicel p32).
+ Chiralcel OD [2080], matching CARBINOXAMINE (Daicel p21).
+ Chiralcel OD [2110], matching 2 BUTYL PHENYL CARBAMATE (Daicel p46).

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule
2-(1-HYDROXYETHYL)NAPHTHALENE....
+ Chiralcel OB [419], matching 1-PHENYLETHANOL (Daicel p42).
+ Chiralcel OD [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel CA-1 [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OJ [465], matching FENOPROFEN (Daicel p27).
+ Chiralcel OB [495], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OJ [567], matching KETOPROFEN (Daicel p27).
+ Chiralcel OB [602], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OD [625], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralcel OB [668], matching 2 ACETOXY-3 PHENYLPROPANE (Daicel p45).
+ Chiralcel OD [703], matching BENZOIN (Daicel p44).
+ Chiralcel OF [703], matching BENZOIN (Daicel p44).
+ Chiralpak OP [703], matching BENZOIN (Daicel p44).
+ Chiralcel OD [709], matching BENZYL PHENYL CARBINOL (Daicel p46).
+ Chiralcel OK [719], matching METHYL MANDELATE (Daicel p44).
+ Chiralcel OJ [818], matching TIAPROFENIC ACID ME ESTER (Daicel p28).
+ Crownpak CR [882], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralpak WH [904], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [904], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OB [967], matching 1,3-DIACETOXY-BUTANE (Daicel p45).
+ Chiralcel OD [970], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OJ [980], matching METHYL2-PHENYLPROPIONATE (Daicel p28).
+ Chiralcel OD [981], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [981], matching PERISOXAL (Daicel p20).
+ Chiralcel OD [1003], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OD [1006], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [1006], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Chiralcel OB [1018], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
+ Crownpak CR [1047], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [1047], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [1047], matching D-PHENYLALANINE (Daicel p34).
+ Chiralcel OB [1052], matching 1 METHYL 3 PHENYL PROPYL BENZOATE (Daicel p45).
+ Chiralpak WH [1123], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1123], matching N-CBZ-VALINE (Daicel p40).
+ Crownpak CR [1168], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Chiralcel OB [1196], matching ALPHA METHYL PHENETHYL BENZOATE (Daicel p45).
+ Crownpak CR [1200], matching PHENYLGLYCINE (Daicel p34).
+ Chiralcel OD [1210], matching METOPROLOL (Daicel p20).
+ Chiralcel OB [1210], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE (Daicel p29).
+ Chiralpak WM [1216], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [1216], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1216], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [1216], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1243], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Crownpak CR [1251], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID (Daicel p35).
+ Chiralpak WH [1265], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1265], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OJ [1282], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OK [1282], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Crownpak CR [1290], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Crownpak CR [1314], matching D-(+)-TRYPTOPHAN (Daicel p38).

+ Chiralpak WH [1314], matching D (+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WE [1318], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1318], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1322], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1322], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1322], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1322], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralcel OB [1336], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralpak OT [1349], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Crownpak CR [1417], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1417], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1417], matching LYSINE {Daicel p38}.
+ Crownpak CR [1432], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [1464], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1464], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1464], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1464], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1473], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1473], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1473], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1490], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Crownpak CR [1559], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1559], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OJ [1767], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1767], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OB [1788], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Crownpak CR [1854], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [2080], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OD [2110], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule N-(PHENYLETHYL)ACETAMIDE....
+ Chiralcel OD [356], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [366], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [464], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [507], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OJ [557], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OB [599], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel CA-1 [642], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [675], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralpak OT [765], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OD [774], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [774], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralpak WH [821], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OJ [830], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [890], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [892], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Crownpak CR [896], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralcel OK [899], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [906], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WM [907], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [907], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [907], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [907], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [950], matching 2 AMINO-1 PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [965], matching 4-AMINO-3 (P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralcel OJ [1018], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1018], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1018], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1018], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1018], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1018], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1018], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Crownpak CR [1036], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1036], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1036], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1036], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [1046], matching 2-AMINO 1 HEXANOL {Daicel p32}.
+ Chiralcel OD [1052], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OJ [1053], matching FENOPROFEN {Daicel p27}.
+ Crownpak CR [1071], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1071], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1071], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [1095], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1095], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1095], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1095], matching DL-VALINE {Daicel p33}.
+ Chiralcel OB [1102], matching N,O-DIBENZOYL2 AMINOBUTANOL {Daicel p46}.
+ Crownpak CR [1105], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralpak WE [1108], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1108], matching DL-LEUCINE {Daicel p33}.
+ Chiralcel OD [1186], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Crownpak CR [1238], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1238], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1238], matching ALANINE {Daicel p33}.
+ Chiralcel OJ [1243], matching KETOPROFEN {Daicel p27}.
+ Crownpak CR [1252], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [1255], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1255], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1255], matching BENZOIN {Daicel p44}.
+ Crownpak CR [1271], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1271], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1271], matching LYSINE {Daicel p38}.
+ Crownpak CR [1282], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1282], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OD [1288], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OD [1439], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1439], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1439], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1439], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak WH [1483], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1483], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralpak WH [1682], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1682], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OJ [1696], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1696], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralpak WH [1824], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1824], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1998], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1998], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [2013], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [2021], matching PROPRANOLOL {Daicel p20}.
+ Chiralpak WH [2070], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [2268], matching METOPROLOL {Daicel p20}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1,3 BUTANEDIOL DIBENZOATE....
+ Chiralcel OB [92], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [296], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [380], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [520], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OB [658], matching METHYL 3 BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [664], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [892], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [913], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OJ [961], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralcel OB [1011], matching 1,3-DIACETOXY BUTANE {Daicel p45}.
+ Chiralcel OB [1031], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1048], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1048], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OJ [1057], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.

+ Chiralcel OK [1057], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1062], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OJ [1127], matching ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE
{Daicel p29}.
+ Chiralcel OB [1186], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1267], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1267], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1305], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1353], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralpak WH [1409], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1409], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1434], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1434], matching PERISOXAL {Daicel p20}.
+ Chiralpak WH [1509], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OB [1575], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Crownpak CR [1595], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Crownpak CR [1701], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1701], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1701], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1736], matching WARFARINE {Daicel p39}.
+ Crownpak CR [1783], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1783], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1783], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1807], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1807], matching L METHIONINE {Daicel p36}.
+ Chiralpak WH [1807], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1807], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1838], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1838], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1838], matching ALANINE {Daicel p33}.
+ Crownpak CR [1845], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WE [1923], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1923], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1924], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1924], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [1926], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1926], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1926], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1926], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [2016], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [2068], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [2068], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [2068], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [2068], matching DL-VALINE {Daicel p33}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1,4 PENTYL DIBENZOATE....
+ Chiralcel OB [70], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [92], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [526], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [666], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [761], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [804], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [810], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [828], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [986], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralpak WH [1054], matching N CBZ ALANINE {Daicel p40}.
+ Chiralcel OD [1054], matching N CBZ ALANINE {Daicel p40}.
+ Chiralcel OB [1158], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [1178], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1201], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OJ [1204], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1204], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1208], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OJ [1273], matching ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE
{Daicel p29}.
+ Chiralpak WH [1273], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1273], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OB [1332], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1415], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1415], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1451], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralpak WH [1515], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1521], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1521], matching PERISOXAL {Daicel p20}.
+ Chiralcel OB [1721], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Crownpak CR [1741], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Crownpak CR [1847], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1847], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1847], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1882], matching WARFARINE {Daicel p39}.
+ Crownpak CR [1929], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1929], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1929], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1953], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1953], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1953], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1953], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1984], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1984], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1984], matching ALANINE {Daicel p33}.
+ Crownpak CR [1991], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WE [2069], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [2069], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [2071], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [2071], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [2072], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [2072], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [2072], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [2072], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [2163], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [2214], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [2214], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [2214], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [2214], matching DL-VALINE {Daicel p33}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1,5 HEXYL DIBENZOATE....
+ Chiralcel OB [70], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [296], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [676], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [708], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [816], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [887], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [954], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [960], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [1014], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralcel OB [1130], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralpak WH [1298], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1298], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OB [1407], matching 1,3-DIACETOXY BUTANE {Daicel p45}.
+ Chiralcel OB [1427], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [1353], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1353], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1357], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OJ [1422], matching ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE
{Daicel p29}.
+ Chiralcel OB [1482], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1517], matching N-CBZ-VALINE {Daicel p40}.

+ Chiralcel OD [1517], matching N-CBZ VALINE {Daicel p40}.
+ Chiralcel OD [1601], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralpak WH [1659], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1659], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OB [1870], matching METHYL
2-[P-(5'-CHLORO 2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralpak WH [1885], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Crownpak CR [1891], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Crownpak CR [1997], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1997], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1997], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [2032], matching WARFARINE {Daicel p39}.
+ Crownpak CR [2078], matching LYSINE {Daicel p38}.
+ Chiralpak WM [2078], matching LYSINE {Daicel p38}.
+ Chiralpak WE [2078], matching LYSINE {Daicel p38}.
+ Chiralpak WM [2103], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [2103], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [2103], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [2103], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [2134], matching ALANINE {Daicel p33}.
+ Chiralpak WH [2134], matching ALANINE {Daicel p33}.
+ Chiralpak WM [2134], matching ALANINE {Daicel p33}.
+ Crownpak CR [2141], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WE [2219], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [2219], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [2220], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [2220], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [2222], matching L ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [2222], matching L ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [2222], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [2222], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [2312], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [2364], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [2364], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [2364], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [2364], matching DL-VALINE {Daicel p33}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 3 PHENYL BUTYL BENZOATE....
+ Chiralcel OJ [619], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralpak OT [643], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OB [798], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [809], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OB [938], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralpak OT [952], matching ETHYL 2,3 DIPHENYL PROPANOATE {Daicel p46}.
+ Chiralcel OD [1112], matching ALPRENOLOL {Daicel p20}.
+ Chiralpak OT [1122], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OB [1150], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OJ [1163], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1163], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Crownpak CR [1197], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralcel OB [1235], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [1260], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Crownpak CR [1328], matching 2-AMINO-3-METHYLBUTANE {Daicel p32}.
+ Chiralcel OB [1370], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [1373], matching PINDOLOL {Daicel p20}.
+ Chiralcel OB [1445], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1563], matching 2 AMINO BUTANE {Daicel p32}.
+ Chiralcel OD [1592], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OD [1614], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1614], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OB [1788], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1788], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1788], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1788], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1788], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1788], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1937], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OD [2209], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [2209], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [2209], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [2209], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule 1 METHYL 3 PHENYL PROPYL
BENZOATE....
+ Chiralcel OB [141], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralpak OT [296], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OB [380], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [526], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [647], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [652], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [654], matching METHYL 3 BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [676], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [689], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OB [807], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OD [930], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [938], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [941], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [941], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1007], matching 1,3 DIACETOXY BUTANE {Daicel p45}.
+ Chiralcel OJ [1018], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [1056], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1056], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1056], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1056], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1056], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1056], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1056], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1084], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OB [1089], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1181], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralcel OJ [1218], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [1225], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [1241], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1241], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OB [1301], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1315], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1315], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1315], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OB [1320], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralpak WH [1368], matching WARFARINE {Daicel p39}.
+ Chiralpak WH [1458], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1458], matching N-CBZ-ALANINE {Daicel p40}.
+ Crownpak CR [1470], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralcel OD [1476], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1476], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1476], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1476], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak WM [1495], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1495], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1495], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1495], matching L-METHIONINE {Daicel p36}.

+ Chiralcel OB [1518], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE [Daicel p29].
+ Chiralpak WE [1537], matching DL-LEUCINE [Daicel p33].
+ Chiralpak WM [1537], matching DL-LEUCINE [Daicel p33].
+ Crownpak CR [1540], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WH [1540], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WM [1540], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WE [1540], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WH [1677], matching N-CBZ-VALINE [Daicel p40].
+ Chiralcel OD [1677], matching N-CBZ-VALINE [Daicel p40].
+ Crownpak CR [1682], matching DL-VALINE [Daicel p33].
+ Chiralpak WH [1682], matching DL-VALINE [Daicel p33].
+ Chiralpak WM [1682], matching DL-VALINE [Daicel p33].
+ Chiralpak WE [1682], matching DL-VALINE [Daicel p33].
+ Crownpak CR [1745], matching LYSINE [Daicel p38].
+ Chiralpak WM [1745], matching LYSINE [Daicel p38].
+ Chiralpak WE [1745], matching LYSINE [Daicel p38].
+ Crownpak CR [1801], matching ALANINE [Daicel p33].
+ Chiralpak WH [1801], matching ALANINE [Daicel p33].
+ Chiralpak WM [1801], matching ALANINE [Daicel p33].
+ Chiralpak WH [1819], matching N-CBZ-NOR-LEUCINE [Daicel p40].
+ Chiralcel OD [1819], matching N-CBZ-NOR-LEUCINE [Daicel p40].
+ Crownpak CR [1887], matching L-ORNITHINE [Daicel p38].
+ Chiralpak WM [1887], matching L-ORNITHINE [Daicel p38].
+ Chiralpak WH [2044], matching N-CBZ-PHENYLALANINE [Daicel p40].

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE....
+ Chiralcel OB [264], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
[Daicel p29].
+ Chiralcel OJ [582], matching TIAPROFENIC ACID ME ESTER [Daicel p28].
+ Chiralcel OB [972], matching METHYL-3-BENZOYLOXYPROPANOATE [Daicel p45].
+ Chiralcel OB [976], matching PENTYNYL 3 BENZOATE [Daicel p45].
+ Chiralcel OJ [1013], matching METHYL 2 PHENOXYPROPIONATE [Daicel p29].
+ Chiralcel OK [1013], matching METHYL 2 PHENOXYPROPIONATE [Daicel p29].
+ Chiralpak WH [1060], matching WARFARINE [Daicel p39].
+ Chiralcel OB [1122], matching BUTYNYL 3 BENZOATE [Daicel p45].
+ Chiralpak OT [1214], matching 2 ETHYL PHENYL BENZOATE [Daicel p46].
+ Chiralcel OB [1378], matching ALPHA METHYL PHENETHYL BENZOATE [Daicel p45].
+ Chiralcel OB [1443], matching N,O-DIBENZOYL2-AMINOBUTANOL [Daicel p46].
+ Chiralcel OB [1518], matching 1 METHYL 3 PHENYL PROPYL BENZOATE [Daicel p45].
+ Chiralcel OB [1590], matching 1,3 DIACETOXY BUTANE [Daicel p45].
+ Chiralcel OB [1596], matching 1,3 BUTANEDIOL DIBENZOATE [Daicel p45].
+ Chiralcel OB [1743], matching 1,4 PENTYL DIBENZOATE [Daicel p45].
+ Crownpak CR [1807], matching ALANINE [Daicel p33].
+ Chiralpak WH [1807], matching ALANINE [Daicel p33].
+ Chiralpak WM [1807], matching ALANINE [Daicel p33].
+ Chiralcel OD [1839], matching 2-PHENYLPROPIONIC ACID [Daicel p27].
+ Chiralcel OB [1842], matching 2-ACETOXY-3-PHENYLPROPANE [Daicel p45].
+ Chiralpak WH [1878], matching N-ACTEYL VALINE [Daicel p39].
+ Chiralcel OB [1892], matching 1,5 HEXYL DIBENZOATE [Daicel p45].
+ Crownpak CR [1992], matching 4 AMINO-3-(P CHLOROPHENYL)BUTYRIC ACID
[Daicel p45].
+ Crownpak CR [2001], matching LYSINE [Daicel p38].
+ Chiralpak WM [2001], matching LYSINE [Daicel p38].
+ Chiralpak WE [2001], matching LYSINE [Daicel p38].
+ Crownpak CR [2026], matching DL-VALINE [Daicel p33].
+ Chiralpak WH [2026], matching DL-VALINE [Daicel p33].
+ Chiralpak WM [2026], matching DL-VALINE [Daicel p33].
+ Chiralpak WE [2026], matching DL-VALINE [Daicel p33].
+ Crownpak CR [2144], matching L-ORNITHINE [Daicel p38].
+ Chiralpak WM [2144], matching L-ORNITHINE [Daicel p38].
+ Crownpak CR [2168], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WH [2168], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WM [2168], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WE [2168], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WE [2171], matching DL-LEUCINE [Daicel p33].
+ Chiralpak WM [2171], matching DL-LEUCINE [Daicel p33].
+ Crownpak CR [2250], matching PHENYLGLYCINE [Daicel p34].
+ Chiralpak WM [2287], matching L-METHIONINE [Daicel p36].
+ Crownpak CR [2287], matching L METHIONINE [Daicel p36].
+ Chiralpak WH [2287], matching L-METHIONINE [Daicel p36].
+ Chiralpak WE [2287], matching L-METHIONINE [Daicel p36].
+ Crownpak CR [2394], matching D-PHENYLALANINE [Daicel p34].
+ Chiralpak WH [2394], matching D-PHENYLALANINE [Daicel p34].
+ Chiralpak WM [2394], matching D-PHENYLALANINE [Daicel p34].
+ Chiralcel OJ [2474], matching FENOPROFEN [Daicel p27].
+ Chiralcel OJ [2514], matching KETOPROFEN [Daicel p27].
+ Chiralpak WH [2666], matching N-CBZ-ALANINE [Daicel p40].
+ Chiralcel OD [2666], matching N-CBZ-ALANINE [Daicel p40].
+ Chiralpak WM [2885], matching N-CBZ-VALINE [Daicel p40].
+ Chiralcel OD [2885], matching N-CBZ-VALINE [Daicel p40].
+ Chiralpak WH [3027], matching N-CBZ-NOR-LEUCINE [Daicel p40].
+ Chiralcel OD [3027], matching N-CBZ-NOR-LEUCINE [Daicel p40].
+ Chiralpak WH [3252], matching N-CBZ-PHENYLALANINE [Daicel p40].

****

*** Testing for suggestions for column Chiralcel OB ....
= Ordered column suggestions [column] for molecule METHYL
2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE....
+ Chiralcel OB [264], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE [Daicel p29].
+ Chiralcel OJ [423], matching TIAPROFENIC ACID ME ESTER [Daicel p28].
+ Chiralcel OJ [718], matching METHYL 2 PHENOXYPROPIONATE [Daicel p29].
+ Chiralcel OK [718], matching METHYL 2 PHENOXYPROPIONATE [Daicel p29].
+ Chiralcel OB [727], matching METHYL-3-BENZOYLOXYPROPANOATE [Daicel p45].
+ Chiralcel OB [778], matching PENTYNYL 3 BENZOATE [Daicel p45].
+ Chiralcel OJ [786], matching KETOPROFEN [Daicel p27].
+ Chiralcel OB [925], matching BUTYNYL 3 BENZOATE [Daicel p45].
+ Chiralpak OT [1016], matching 2 ETHYL PHENYL BENZOATE [Daicel p46].
+ Chiralcel OD [1020], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
[Daicel p43].
+ Chiralcel OG [1020], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
[Daicel p43].
+ Chiralpak OT [1020], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
[Daicel p43].
+ Chiralcel CA-1 [1020], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
[Daicel p43].
+ Chiralcel OB [1130], matching 1,5 HEXYL DIBENZOATE [Daicel p45].
+ Chiralcel OD [1174], matching 2PHENOXYPROPIONIC ACID [Daicel p29].
+ Chiralcel OF [1174], matching 2PHENOXYPROPIONIC ACID [Daicel p26].
+ Chiralcel OB [1180], matching ALPHA METHYL PHENETHYL BENZOATE [Daicel p45].
+ Chiralcel OJ [1185], matching ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE [Daicel
p29].
+ Chiralcel OB [1201], matching 1,4 PENTYL DIBENZOATE [Daicel p45].
+ Chiralcel OJ [1210], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OB [1210], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OK [1210], matching 2-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OB [1210], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OC [1210], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OJ [1210], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OD [1210], matching 1-(1-HYDROXYETHYL)NAPHTHALENE [Daicel p43].
+ Chiralcel OB [1320], matching 1 METHYL 3 PHENYL PROPYL BENZOATE [Daicel p45].
+ Chiralpak WH [1329], matching WARFARINE [Daicel p39].
+ Chiralcel OB [1353], matching 1,3 BUTANEDIOL DIBENZOATE [Daicel p45].
+ Chiralcel OB [1510], matching 1,3-DIACETOXY-BUTANE [Daicel p45].
+ Crownpak CR [1618], matching ALANINE [Daicel p33].
+ Chiralpak WH [1618], matching ALANINE [Daicel p33].
+ Chiralpak WM [1618], matching ALANINE [Daicel p33].
+ Chiralcel OD [1650], matching 2-PHENYLPROPIONIC ACID [Daicel p27].
+ Chiralcel OB [1762], matching 2-ACETOXY-3-PHENYLPROPANE [Daicel p45].
+ Chiralpak WH [1764], matching LACTIC ACID [Daicel p27].
+ Crownpak CR [1837], matching DL-VALINE [Daicel p33].
+ Chiralpak WH [1837], matching DL-VALINE [Daicel p33].
+ Chiralpak WM [1837], matching DL-VALINE [Daicel p33].
+ Chiralpak WE [1837], matching DL-VALINE [Daicel p33].

+ Crownpak CR [1912], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Crownpak CR [1979], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1979], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1979], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1979], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1982], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1982], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [2060], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [2077], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [2077], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [2077], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [2077], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [2088], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [2088], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [2204], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [2204], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [2204], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [2232], matching LYSINE {Daicel p38}.
+ Chiralpak WM [2232], matching LYSINE {Daicel p38}.
+ Chiralpak WE [2232], matching LYSINE {Daicel p38}.
+ Chiralcel OJ [2285], matching FENOPROFEN {Daicel p27}.
+ Chiralpak WH [2476], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [2476], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralpak WH [2695], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [2695], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [2847], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [2847], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralpak WH [3063], matching N-CBZ-PHENYLALANINE {Daicel p40}.

****

*** Testing for suggestions for column Chiralcel OB ....
- Ordered column suggestions [column] for molecule PENTYNYL 3 BENZOATE....
+ Chiralcel OB [155], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [323], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [368], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [395], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [395], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [512], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OD [587], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OB [652], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [778], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OB [951], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [976], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OB [980], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [1002], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1031], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1178], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1249], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1249], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1249], matching ALANINE {Daicel p33}.
+ Crownpak CR [1319], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1319], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1319], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1319], matching DL-VALINE {Daicel p33}.
+ Chiralcel OB [1327], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1411], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Chiralpak WH [1440], matching WARFARINE {Daicel p39}.
+ Crownpak CR [1461], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1461], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1461], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1461], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1464], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1464], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1542], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [1580], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1580], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1580], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1580], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1686], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1686], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1686], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OJ [1713], matching FENOPROFEN {Daicel p27}.
+ Crownpak CR [1721], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1721], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WH [1783], matching 2 AMINO-3 PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralpak WH [1855], matching N-ACETYL-L PHENYLALANINE {Daicel p32}.
+ Crownpak CR [1865], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1865], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1865], matching LYSINE {Daicel p38}.
+ Chiralcel OJ [1869], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [2076], matching N-AMIDE-L PHENYLALANINE {Daicel p41}.
+ Chiralpak WH [2109], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [2109], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralpak WH [2328], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [2328], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [2470], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [2470], matching N-CBZ-NOR LEUCINE {Daicel p40}.
+ Chiralpak WH [2695], matching N-CBZ-PHENYLALANINE {Daicel p40}.

****

*** Testing for suggestions for column Chiralcel OB ....
- Ordered column suggestions [column] for molecule BUTYNYL 3 BENZOATE....
+ Chiralcel OB [155], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OT [523], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [546], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [546], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [562], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [667], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OD [710], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [710], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OB [807], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [925], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OB [1010], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [1042], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [1095], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1095], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1095], matching ALANINE {Daicel p33}.
+ Chiralcel OB [1122], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Crownpak CR [1134], matching DL-TYROSINE {Daicel p35}.
+ Chiralcel OD [1157], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1186], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1313], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1313], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1313], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1313], matching DL-VALINE {Daicel p33}.
+ Chiralcel OB [1332], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1359], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Crownpak CR [1374], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Crownpak CR [1456], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1456], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1456], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1456], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1459], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1459], matching DL-LEUCINE {Daicel p33}.
+ Chiralcel OB [1482], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1518], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1518], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1518], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [1537], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [1541], matching ARGININE {Daicel p32}.
+ Chiralpak WE [1541], matching ARGININE {Daicel p32}.

+ Chiralpak WM [1541], matching ARGININE (Daicel p39).
+ Chiralpak WM [1575], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [1575], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1575], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [1575], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1595], matching WARFARINE (Daicel p39).
+ Crownpak CR [1613], matching LYSINE (Daicel p38).
+ Chiralpak WM [1613], matching LYSINE (Daicel p38).
+ Chiralpak WE [1613], matching LYSINE (Daicel p38).
+ Chiralpak WH [1658], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Crownpak CR [1683], matching CITRULLINE (Daicel p37).
+ Crownpak CR [1715], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [1715], matching L-ORNITHINE (Daicel p38).
+ Chiralcel OJ [1868], matching FENOPROFEN (Daicel p27).
+ Chiralcel OJ [2024], matching KETOPROFEN (Daicel p27).
+ Chiralpak WH [2066], matching N-AMIDE-L-PHENYLALANINE (Daicel p41).
+ Chiralpak WH [2264], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [2264], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralpak WH [2483], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [2483], matching N-CBZ-VALINE (Daicel p40).
+ Chiralpak WH [2544], matching N-CBZ-PHENYLALANINE (Daicel p40).
+ Chiralpak WH [2625], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [2625], matching N-CBZ-NOR-LEUCINE (Daicel p40).

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule MANDELIC ACID....
+ Chiralcel OB [133], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Chiralcel OB [200], matching 1-PHENYL-1,3-PROPANEDIOL (Daicel p44).
+ Chiralcel CA-1 [325], matching MANDELAMIDE (Daicel p44).
+ Crownpak CR [344], matching PHENYLGLYCINE (Daicel p34).
+ Chiralpak WM [355], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [355], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [355], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [355], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [406], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralpak WH [443], matching LACTIC ACID (Daicel p27).
+ Chiralcel OK [456], matching METHYL MANDELATE (Daicel p44).
+ Crownpak CR [476], matching 2-AMINO 2-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [507], matching L ISOLEUCINE (Daicel p34).
+ Chiralpak WH [507], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [507], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [507], matching L-ISOLEUCINE (Daicel p34).
+ Crownpak CR [566], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [566], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [566], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [566], matching DL-VALINE (Daicel p33).
+ Chiralcel OD [625], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralcel OB [626], matching 1-PHENYLETHANOL (Daicel p42).
+ Chiralcel OB [642], matching 1-PHENYL-PROPAN 1-OL (Daicel p42).
+ Crownpak CR [652], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Crownpak CR [711], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Crownpak CR [737], matching LYSINE (Daicel p38).
+ Chiralpak WM [737], matching LYSINE (Daicel p38).
+ Chiralpak WE [737], matching LYSINE (Daicel p38).
+ Crownpak CR [753], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [753], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WH [754], matching N-ACETYL-L-PHENYLALANINE (Daicel p49).
+ Chiralcel OB [757], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralpak WH [798], matching N ACTEYL VALINE (Daicel p49).
+ Crownpak CR [930], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [930], matching HISTIDINE (Daicel p38).
+ Crownpak CR [935], matching L-GLUTAMIC ACID (Daicel p36).
+ Chiralcel OD [958], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [958], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Crownpak CR [1008], matching 1-PHENYLETHYLAMINE (Daicel p32).
+ Crownpak CR [1072], matching SERINE (Daicel p35).
+ Chiralpak WH [1072], matching SERINE (Daicel p35).
+ Chiralpak OT [1148], matching 1-BROMO ETHYLBENZENE (Daicel p46).
+ Crownpak CR [1239], matching THREONINE (Daicel p35).
+ Chiralpak WH [1239], matching THREONINE (Daicel p35).
+ Crownpak CR [1265], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WH [1265], matching ASPARTIC ACID (Daicel p37).
+ Chiralpak WM [1265], matching ASPARTIC ACID (Daicel p37).
+ Crownpak CR [1280], matching CITRULLINE (Daicel p37).
+ Crownpak CR [1462], matching ASPARAGINE (Daicel p37).
+ Chiralcel OJ [1532], matching TRIADIMEFON (Daicel p22).
+ Crownpak CR [1643], matching L-GLUTAMINE (Daicel p36).
+ Chiralcel OD [1692], matching PRACTOLOL (Daicel p20).
+ Chiralpak WH [1706], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Chiralcel OD [1706], matching N-CBZ-ASPARAGINE (Daicel p40).
+ Crownpak CR [1949], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [1949], matching D-(+)-TRYPTOPHAN (Daicel p38).

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule 2-PHENYLPROPIONIC ACID....
+ Chiralcel OB [133], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OB [206], matching 1-PHENYLETHANOL (Daicel p42).
+ Chiralcel OB [282], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OB [298], matching 2-ACETOXY-3-PHENYLPROPANE (Daicel p45).
+ Chiralcel OD [334], matching 2PHENOXYPROPIONIC ACID (Daicel p29).
+ Chiralcel OF [334], matching 2PHENOXYPROPIONIC ACID (Daicel p26).
+ Chiralcel OB [349], matching 1,3-DIACETOXY-BUTANE (Daicel p45).
+ Chiralcel OB [356], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
+ Chiralcel OJ [456], matching METHYL2-PHENYLPROPIONATE (Daicel p28).
+ Crownpak CR [472], matching 1-PHENYLETHYLAMINE (Daicel p32).
+ Chiralcel OK [505], matching METHYL MANDELATE (Daicel p44).
+ Chiralpak OT [524], matching 1-BROMO ETHYLBENZENE (Daicel p46).
+ Crownpak CR [524], matching PHENYLGLYCINE (Daicel p34).
+ Chiralpak WM [535], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [535], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [535], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [535], matching L-METHIONINE (Daicel p36).
+ Chiralcel OJ [625], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [625], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OK [625], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [625], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OC [625], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [625], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [625], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralpak WH [625], matching MANDELIC ACID (Daicel p27).
+ Chiralcel OD [625], matching MANDELIC ACID (Daicel p27).
+ Chiralpak OT [635], matching 2 ETHYL PHENYL BENZOATE (Daicel p46).
+ Crownpak CR [669], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralcel OJ [677], matching FENOPROFEN (Daicel p27).
+ Crownpak CR [678], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [681], matching L ISOLEUCINE (Daicel p34).
+ Chiralpak WH [681], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [681], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [681], matching L-ISOLEUCINE (Daicel p34).
+ Crownpak CR [695], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [695], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [695], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WE [732], matching DL-LEUCINE (Daicel p33).
+ Chiralpak WM [732], matching DL-LEUCINE (Daicel p33).
+ Crownpak CR [739], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [739], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [739], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [739], matching DL-VALINE (Daicel p33).
+ Chiralcel OB [757], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Chiralcel OJ [789], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OK [789], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
+ Chiralcel OB [790], matching ALPHA METHYL PHENETHYL BENZOATE (Daicel p45).
+ Chiralcel OD [792], matching BENZYL PHENYL CARBINOL (Daicel p46).
+ Crownpak CR [827], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
(Daicel p35).
+ Chiralcel OD [862], matching BENZOIN (Daicel p44).

+ Chiralcel OF [862], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [862], matching BENZOIN {Daicel p44}.
+ Crownpak CR [863], matching ALANINE {Daicel p33}.
+ Chiralpak WH [863], matching ALANINE {Daicel p33}.
+ Chiralpak WM [863], matching ALANINE {Daicel p33}.
+ Chiralcel OJ [868], matching KETOPROFEN {Daicel p27}.
+ Crownpak CR [879], matching LYSINE {Daicel p38}.
+ Chiralpak WM [879], matching LYSINE {Daicel p38}.
+ Chiralpak WE [879], matching LYSINE {Daicel p38}.
+ Crownpak CR [926], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [926], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OB [930], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [948], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel CA-1 [950], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [1009], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OD [1046], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1046], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1046], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1046], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak WH [1108], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1108], matching N-CBZ-ALANINE {Daicel p40}.
+ Crownpak CR [1131], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1131], matching HISTIDINE {Daicel p38}.
+ Chiralcel OB [1164], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1260], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Chiralcel OB [1305], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [1326], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1326], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OB [1451], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [1469], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1469], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Crownpak CR [1481], matching CITRULLINE {Daicel p37}.
+ Chiralcel OB [1601], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [1604], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1604], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1619], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1628], matching PROPRANOLOL {Daicel p20}.
+ Chiralpak WH [1694], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OB [1697], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OB [1839], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OD [1910], matching METOPROLOL {Daicel p20}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule 2 BUTYL PHENYL CARBAMATE....
+ Chiralcel OB [587], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [912], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [1052], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [1079], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OD [1200], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OB [1206], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Crownpak CR [1271], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [1306], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [1443], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1651], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [1727], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1760], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralcel OD [1834], matching GUAIFENESINE {Daicel p44}.
+ Chiralpak WH [1898], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [1902], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OK [1950], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [1972], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Crownpak CR [1983], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OJ [2067], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [2067], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OB [2110], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [2110], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [2110], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [2110], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [2110], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [2110], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [2110], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Crownpak CR [2114], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [2237], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OD [2307], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [2307], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [2307], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [2549], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [2549], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [2549], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [2549], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OD [3044], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [3044], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [3116], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [3134], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [3320], matching METOPROLOL {Daicel p20}.

****

*** Testing for suggestions for column Chiralcel OD ....

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule PRACTOLOL....
+ Chiralcel OD [369], matching PINDOLOL {Daicel p20}.
+ Chiralcel OD [616], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OD [652], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [681], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [739], matching PROPRANOLOL {Daicel p20}.
+ Crownpak CR [1260], matching ARGININE {Daicel p39}.
+ Chiralpak WE [1260], matching ARGININE {Daicel p39}.
+ Chiralpak WM [1260], matching ARGININE {Daicel p39}.
+ Chiralpak WH [1550], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Chiralcel OB [1592], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Crownpak CR [1653], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1653], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [1694], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [1712], matching L-GLUTAMINE {Daicel p36}.
+ Chiralpak WH [1715], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [1715], matching MANDELIC ACID {Daicel p27}.
+ Chiralpak WH [1719], matching N-ACTEYL VALINE {Daicel p39}.
+ Crownpak CR [1735], matching CITRULLINE {Daicel p37}.
+ Crownpak CR [1747], matching ASPARAGINE {Daicel p37}.
+ Chiralcel OJ [1763], matching TRIADIMEFON {Daicel p22}.
+ Chiralpak WH [1882], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1882], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Crownpak CR [1904], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OD [1972], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1972], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Crownpak CR [2051], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [2051], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [2051], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [2106], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [2189], matching SERINE {Daicel p35}.
+ Chiralpak WH [2189], matching SERINE {Daicel p35}.
+ Crownpak CR [2357], matching THREONINE {Daicel p35}.
+ Chiralpak WH [2357], matching THREONINE {Daicel p35}.
+ Crownpak CR [3067], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [3067], matching D-(+)-TRYPTOPHAN {Daicel p38}.

```
************************************************************
*** Testing for suggestions for column Chiralcel OD ....
 = Ordered column suggestions [column] for molecule BENZOIN....
 + Chiralcel OD [10], matching BENZYL PHENYL CARBINOL {Daicel p46}.
 + Chiralcel OK [544], matching METHYL MANDELATE {Daicel p44}.
 + Chiralcel OB [581], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
 + Chiralcel OB [656], matching 1-PHENYLETHANOL {Daicel p42}.
 + Crownpak CR [663], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
 + Chiralcel OD [696], matching PERISOXAL {Daicel p20}.
 + Chiralcel OF [696], matching PERISOXAL {Daicel p20}.
 + Chiralcel OJ [703], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
 + Chiralcel OB [703], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
 + Chiralcel OK [703], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
 + Chiralcel OB [703], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
 + Chiralcel OC [703], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
 + Chiralcel OJ [703], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
 + Chiralcel OD [703], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
 + Chiralpak OT [728], matching ETHYL 2,3 DIPHENYL PROPANOATE {Daicel p46}.
 + Crownpak CR [790], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
 + Chiralcel OB [855], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
 + Chiralcel OD [862], matching ALPRENOLOL {Daicel p20}.
 + Chiralcel OB [865], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
 + Chiralcel OD [872], matching PROPRANOLOL {Daicel p20}.
 + Chiralcel OD [897], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
 + Chiralpak OT [1031], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
 + Chiralcel OD [1064], matching METOPROLOL {Daicel p20}.
 + Chiralcel OB [1107], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
 + Crownpak CR [1113], matching D-PHENYLALANINE {Daicel p34}.
 + Chiralpak WH [1113], matching D-PHENYLALANINE {Daicel p34}.
 + Chiralpak WM [1113], matching D-PHENYLALANINE {Daicel p34}.
 + Chiralcel OD [1121], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         {Daicel p43}.
 + Chiralcel OG [1121], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         {Daicel p43}.
 + Chiralpak OT [1121], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         {Daicel p44}.
 + Chiralcel CA 1 [1121], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         {Daicel p43}.
 + Chiralcel OJ [1141], matching CYCLOPENTOLATE {Daicel p21}.
 + Chiralcel OD [1141], matching CYCLOPENTOLATE {Daicel p21}.
 + Crownpak CR [1168], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
 + Crownpak CR [1200], matching PHENYLGLYCINE {Daicel p34}.
 + Chiralcel OJ [1203], matching FENOPROFEN {Daicel p27}.
 + Crownpak CR [1216], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
         {Daicel p35}.
 + Chiralpak WM [1222], matching L-METHIONINE {Daicel p36}.
 + Crownpak CR [1222], matching L-METHIONINE {Daicel p36}.
 + Chiralpak WH [1222], matching L-METHIONINE {Daicel p36}.
 + Chiralpak WE [1222], matching L-METHIONINE {Daicel p36}.
 + Chiralcel OB [1255], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
 + Chiralpak WH [1299], matching N-CBZ-PHENYLALANINE {Daicel p40}.
 + Chiralcel OJ [1303], matching KETOPROFEN {Daicel p27}.
 + Chiralpak WH [1309], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
 + Chiralpak WH [1323], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
 + Chiralcel OD [1323], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
 + Crownpak CR [1345], matching 2-AMINO-1-HEXANOL {Daicel p32}.
 + Chiralpak WH [1358], matching N-CBZ-VALINE {Daicel p40}.
 + Chiralcel OD [1358], matching N-CBZ-VALINE {Daicel p40}.
 + Chiralpak WE [1376], matching DL-LEUCINE {Daicel p33}.
 + Chiralpak WM [1376], matching DL-LEUCINE {Daicel p33}.
 + Crownpak CR [1377], matching L-ISOLEUCINE {Daicel p34}.
 + Chiralpak WH [1377], matching L-ISOLEUCINE {Daicel p34}.
 + Chiralpak WM [1377], matching L-ISOLEUCINE {Daicel p34}.
 + Chiralpak WE [1377], matching L-ISOLEUCINE {Daicel p34}.
 + Crownpak CR [1391], matching 2-AMINO-1-PENTANOL {Daicel p32}.
 + Crownpak CR [1423], matching DL-VALINE {Daicel p33}.
 + Chiralpak WH [1423], matching DL-VALINE {Daicel p33}.
 + Chiralpak WM [1423], matching DL-VALINE {Daicel p33}.
 + Chiralpak WE [1423], matching DL-VALINE {Daicel p33}.
 + Crownpak CR [1448], matching LYSINE {Daicel p38}.
 + Chiralpak WM [1448], matching LYSINE {Daicel p38}.
 + Chiralpak WE [1448], matching LYSINE {Daicel p38}.
 + Chiralcel OD [1459], matching CARBINOXAMINE {Daicel p21}.
 + Chiralpak WH [1577], matching N-CBZ-ALANINE {Daicel p40}.
 + Chiralcel OD [1577], matching N-CBZ-ALANINE {Daicel p40}.
 + Crownpak CR [1608], matching L-ORNITHINE {Daicel p38}.
 + Chiralpak WM [1608], matching L-ORNITHINE {Daicel p38}.
 + Crownpak CR [1642], matching ALANINE {Daicel p33}.
 + Chiralpak WH [1642], matching ALANINE {Daicel p33}.
 + Chiralpak WM [1642], matching ALANINE {Daicel p33}.
 + Chiralcel OB [1743], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
 + Chiralcel OD [2307], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.

************************************************************
*** Testing for suggestions for column Chiralcel OD ....
 = Ordered column suggestions [column] for molecule CARTEOLOL....
 + Chiralcel OD [616], matching PRACTOLOL {Daicel p20}.
 + Chiralcel OD [817], matching PINDOLOL {Daicel p20}.
 + Chiralcel OD [892], matching ALPRENOLOL {Daicel p20}.
 + Chiralcel OD [918], matching PROPRANOLOL {Daicel p20}.
 + Crownpak CR [1884], matching LYSINE {Daicel p38}.
 + Chiralpak WM [1884], matching LYSINE {Daicel p38}.
 + Chiralpak WE [1884], matching LYSINE {Daicel p38}.
 + Crownpak CR [1934], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
 + Chiralcel OD [1941], matching METOPROLOL {Daicel p20}.
 + Crownpak CR [2027], matching L-ORNITHINE {Daicel p38}.
 + Chiralpak WM [2027], matching L-ORNITHINE {Daicel p38}.
 + Chiralpak WM [2268], matching L-METHIONINE {Daicel p36}.
 + Crownpak CR [2268], matching L-METHIONINE {Daicel p36}.
 + Chiralpak WH [2268], matching L-METHIONINE {Daicel p36}.
 + Chiralpak WE [2268], matching L-METHIONINE {Daicel p36}.
 + Chiralcel OB [2297], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
 + Crownpak CR [2337], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
 + Crownpak CR [2352], matching 2-AMINO 1 HEXANOL {Daicel p32}.
 + Crownpak CR [2368], matching PHENYLGLYCINE {Daicel p34}.
 + Crownpak CR [2384], matching L-ISOLEUCINE {Daicel p34}.
 + Chiralpak WH [2384], matching L-ISOLEUCINE {Daicel p34}.
 + Chiralpak WM [2384], matching L-ISOLEUCINE {Daicel p34}.
 + Chiralpak WE [2384], matching L-ISOLEUCINE {Daicel p34}.
 + Chiralpak WE [2387], matching DL-LEUCINE {Daicel p33}.
 + Chiralpak WM [2387], matching DL-LEUCINE {Daicel p33}.
 + Crownpak CR [2393], matching 2-AMINO-1-PENTANOL {Daicel p32}.
 + Crownpak CR [2425], matching DL-VALINE {Daicel p33}.
 + Chiralpak WH [2425], matching DL-VALINE {Daicel p33}.
 + Chiralpak WM [2425], matching DL-VALINE {Daicel p33}.
 + Chiralpak WE [2425], matching DL-VALINE {Daicel p33}.
 + Crownpak CR [2506], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
         {Daicel p35}.
 + Crownpak CR [2509], matching D-PHENYLALANINE {Daicel p34}.
 + Chiralpak WH [2509], matching D-PHENYLALANINE {Daicel p34}.
 + Chiralpak WM [2509], matching D-PHENYLALANINE {Daicel p34}.
 + Crownpak CR [2644], matching ALANINE {Daicel p33}.
 + Chiralpak WH [2644], matching ALANINE {Daicel p33}.
 + Chiralpak WM [2644], matching ALANINE {Daicel p33}.
 + Chiralpak WM [2706], matching 2-AMINO 3 PHENYLPROPIONAMIDE {Daicel p41}.
 + Crownpak CR [3022], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.

************************************************************
*** Testing for suggestions for column Chiralcel OD ....
 = Ordered column suggestions [column] for molecule CYCLOPENTOLATE....
 + Chiralcel OD [749], matching CARBINOXAMINE {Daicel p21}.
 + Chiralpak OT [790], matching ETHYL 2,3 DIPHENYL PROPANOATE {Daicel p46}.
 + Chiralpak WH [802], matching WARFARINE {Daicel p39}.
 + Chiralcel OD [1141], matching BENZOIN {Daicel p44}.
 + Chiralcel OF [1141], matching BENZOIN {Daicel p44}.
 + Chiralpak OP [1141], matching BENZOIN {Daicel p44}.
 + Chiralcel OB [1161], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
 + Crownpak CR [1187], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
```

+ Chiralcel OD [1207], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OD [1303], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1303], matching PERISOXAL {Daicel p20}.
+ Chiralcel OJ [1342], matching TRIADIMEFON {Daicel p22}.
+ Crownpak CR [1507], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [1518], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Crownpak CR [1522], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OK [1562], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [1629], matching 2 PHENYL PROPAN 1 OL {Daicel p42}.
+ Chiralcel OD [1646], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1664], matching ALPRENOLOL {Daicel p20}.
+ Crownpak CR [1696], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OB [1696], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1730], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralpak WH [1740], matching N-AMIDE L PHENYLALANINE {Daicel p41}.
+ Chiralcel OB [1762], matching 1-PHENYL PROPAN-1-OL {Daicel p42}.
+ Chiralcel OJ [1767], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1767], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1767], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1767], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1767], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1767], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1767], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralpak WH [1815], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OB [1838], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [1850], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [2067], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OD [2199], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [2199], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [2199], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [2199], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule GUAIFENESINE....
+ Chiralcel OB [541], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [668], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [668], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralcel OB [709], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel CA-1 [820], matching MANDELAMIDE {Daicel p44}.
+ Chiralpak WH [1035], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Chiralpak WH [1166], matching GLYCERIC ACID {Daicel p27}.
+ Chiralpak WH [1205], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1205], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Crownpak CR [1333], matching ARGININE {Daicel p39}.
+ Chiralpak WE [1333], matching ARGININE {Daicel p39}.
+ Chiralpak WM [1333], matching ARGININE {Daicel p39}.
+ Crownpak CR [1369], matching SERINE {Daicel p35}.
+ Chiralpak WH [1369], matching SERINE {Daicel p35}.
+ Chiralpak WH [1402], matching N-ACTEYL VALINE {Daicel p39}.
+ Crownpak CR [1524], matching THREONINE {Daicel p35}.
+ Chiralpak WH [1524], matching THREONINE {Daicel p35}.
+ Chiralcel OD [1575], matching PINDOLOL {Daicel p20}.
+ Crownpak CR [1766], matching ASPARAGINE {Daicel p37}.
+ Chiralcel OD [1834], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Crownpak CR [1945], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [1969], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1969], matching HISTIDINE {Daicel p38}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule 1-(1-HYDROXYETHYL)NAPHTHALENE....
+ Chiralcel OB [419], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [421], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OJ [465], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [495], matching 1 PHENYL PROPAN 1 OL {Daicel p42}.
+ Chiralcel OJ [567], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [602], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OD [625], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [668], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [703], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [703], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [703], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [709], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OK [719], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OJ [818], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Crownpak CR [882], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [904], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [904], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OB [967], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [970], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OJ [980], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OD [981], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [981], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1003], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1006], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1006], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OB [1018], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1047], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1047], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1047], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OB [1052], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1123], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1123], matching N-CBZ-VALINE {Daicel p40}.
+ Crownpak CR [1168], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [1196], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Crownpak CR [1200], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralcel OD [1210], matching METOPROLOL {Daicel p20}.
+ Chiralpak OB [1210], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralpak WM [1216], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1216], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1216], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1216], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1243], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralpak OB [1251], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Chiralpak WH [1265], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1265], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OJ [1282], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1282], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Crownpak CR [1290], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [1314], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [1314], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WE [1318], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1318], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1322], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1322], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1322], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1322], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralcel OB [1336], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralpak OT [1349], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Crownpak CR [1417], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1417], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1417], matching LYSINE {Daicel p38}.
+ Crownpak CR [1432], matching 2-AMINO-1-PENTANOL {Daicel p32}.

+ Crownpak CR [1464], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1464], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1464], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1464], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1473], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1473], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1473], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1490], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Crownpak CR [1559], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1559], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OJ [1767], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1767], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OB [1788], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Crownpak CR [1854], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [2080], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OD [2110], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule
9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE....
+ Chiralcel OJ [300], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OJ [359], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [421], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [421], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [421], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [421], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [421], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [421], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [421], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [675], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralcel OB [840], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [916], matching 1-PHENYL-PROPAN 1-OL {Daicel p42}.
+ Chiralcel OD [981], matching PROPRANOLOL {Daicel p20}.
+ Chiralpak WH [1014], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1014], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OB [1020], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OB [1022], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OD [1034], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1046], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1089], matching 2-ACETOXY 3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [1121], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [1121], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1121], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1121], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [1129], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OK [1139], matching METHYL MANDELATE {Daicel p44}.
+ Chiralpak WH [1233], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1233], matching N-CBZ-VALINE {Daicel p40}.
+ Crownpak CR [1303], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [1366], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralpak WH [1375], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1375], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1392], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1392], matching PERISOXAL {Daicel p20}.
+ Chiralcel OJ [1401], matching METHYL2 PHENYLPROPIONATE {Daicel p28}.
+ Chiralcel OD [1424], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1424], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Crownpak CR [1435], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1435], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1435], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OB [1439], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [1476], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Crownpak CR [1556], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [1588], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WH [1601], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralpak WM [1602], matching L METHIONINE {Daicel p36}.
+ Crownpak CR [1602], matching L METHIONINE {Daicel p36}.
+ Chiralpak WH [1602], matching L METHIONINE {Daicel p36}.
+ Chiralpak WE [1602], matching L-METHIONINE {Daicel p36}.
+ Chiralcel OB [1616], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1631], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [1690], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Crownpak CR [1696], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OJ [1701], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1701], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralpak WE [1724], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1724], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1727], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1727], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1727], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1727], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralcel OB [1743], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralpak OT [1769], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Crownpak CR [1783], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1783], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1783], matching LYSINE {Daicel p38}.
+ Crownpak CR [1838], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [1869], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1869], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1869], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1869], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1871], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1871], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1871], matching ALANINE {Daicel p33}.
+ Crownpak CR [1925], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1925], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OJ [2199], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [2199], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OB [2209], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Crownpak CR [2242], matching ERYTHRO-2 AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [2519], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OD [2549], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule CARBINOXAMINE....
+ Chiralpak OT [905], matching ETHYL 2,3 DIPHENYL PROPANOATE {Daicel p46}.
+ Crownpak CR [1110], matching ERYTHRO 2 AMINO 1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OJ [1117], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1117], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralpak WH [1230], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OB [1230], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OD [1239], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.
+ Chiralcel OB [1288], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OJ [1349], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OD [1461], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OD [1529], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1529], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1529], matching BENZOIN {Daicel p44}.
+ Crownpak CR [1548], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [1600], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OK [1611], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [1638], matching 2-AMINO-1 PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [1680], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1704], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralcel OD [1785], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1785], matching PERISOXAL {Daicel p20}.
+ Chiralcel OB [1811], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OD [1858], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1876], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OB [1887], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [2061], matching METOPROLOL {Daicel p20}.
+ Chiralcel OJ [2080], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [2080], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [2080], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.

+ Chiralcel OH [2080], matching 1 (1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [2080], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [2080], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [2080], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [2519], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [2519], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [2519], matching 9 (2 TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [2519], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule SULCONAZOL....

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule BENZYL PHENYL CARBINOL....
+ Chiralcel OD [10], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [10], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [10], matching BENZOIN {Daicel p44}.
+ Chiralcel OB [511], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OH [587], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OK [612], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OH [709], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [709], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [709], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [709], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [709], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OH [709], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [709], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralpak OT [718], matching ETHYL 2,3 DIPHENYL PROPANOATE {Daicel p46}.
+ Crownpak CR [735], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [763], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [763], matching PERISOXAL {Daicel p20}.
+ Chiralcel OH [796], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OD [827], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [845], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [857], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OD [869], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [878], matching PROPRANOLOL {Daicel p20}.
+ Chiralpak OT [1102], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Crownpak CR [1103], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1103], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1103], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OD [1129], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1129], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1129], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1129], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OD [1134], matching METOPROLOL {Daicel p20}.
+ Crownpak CR [1148], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralcel OB [1161], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [1186], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OJ [1207], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [1207], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1207], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralpak WM [1215], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1215], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1215], matching L METHIONINE {Daicel p36}.
+ Chiralpak WE [1215], matching L METHIONINE {Daicel p36}.
+ Crownpak CR [1236], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [1262], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [1267], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WE [1293], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1293], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1294], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1294], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1294], matching L ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1294], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1299], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralpak WH [1348], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1390], matching CARBINOXAMINE {Daicel p21}.
+ Chiralpak WH [1393], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1393], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OJ [1406], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [1427], matching N-CBZ VALINE {Daicel p40}.
+ Chiralcel OD [1427], matching N-CBZ VALINE {Daicel p40}.
+ Crownpak CR [1439], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [1471], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1471], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1471], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1471], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1502], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1502], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1502], matching LYSINE {Daicel p38}.
+ Chiralpak WH [1646], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1646], matching N-CBZ-ALANINE {Daicel p40}.
+ Crownpak CR [1676], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1676], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [1690], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1690], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1690], matching ALANINE {Daicel p33}.
+ Chiralcel OH [1813], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OD [2237], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule PROPRANOLOL....
+ Chiralcel OD [67], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [237], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [357], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [357], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [739], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OD [872], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [872], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [872], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [878], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OD [918], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OD [981], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [981], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [981], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [981], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OJ [1003], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1003], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1003], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1003], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1003], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1003], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1003], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1052], matching FENOPROFEN {Daicel p27}.
+ Crownpak CR [1083], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OK [1146], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OJ [1208], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [1267], matching N CBZ PHENYLALANINE {Daicel p40}.
+ Chiralpak WH [1280], matching N-CBZ-NOR LEUCINE {Daicel p40}.
+ Chiralcel OD [1280], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Crownpak CR [1319], matching D-PHENYLALANINE {Daicel p34}.

+ Chiralpak WH [1319], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [1319], matching D-PHENYLALANINE (Daicel p34).
+ Chiralcel CA-1 [1335], matching MANDELAMIDE (Daicel p44).
+ Chiralcel OB [1346], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OB [1422], matching 1 PHENYLETHANOL (Daicel p42).
+ Chiralpak WH [1447], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1447], matching N-CBZ-VALINE (Daicel p40).
+ Crownpak CR [1448], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [1472], matching PHENYLGLYCINE (Daicel p34).
+ Crownpak CR [1475], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID (Daicel p35).
+ Chiralpak WM [1487], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [1487], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1487], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [1487], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1550], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Crownpak CR [1630], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Chiralcel OB [1631], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OJ [1655], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OD [1655], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralpak WE [1659], matching DL-LEUCINE (Daicel p33).
+ Chiralpak WM [1659], matching DL-LEUCINE (Daicel p33).
+ Crownpak CR [1662], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WH [1662], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [1662], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [1662], matching L-ISOLEUCINE (Daicel p34).
+ Chiralcel OD [1663], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralpak WH [1666], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [1666], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OB [1671], matching 2-ACETOXY-3-PHENYLPROPANE (Daicel p45).
+ Chiralcel OB [1675], matching 1,3-DIACETOXY-BUTANE (Daicel p45).
+ Crownpak CR [1707], matching LYSINE (Daicel p38).
+ Chiralpak WM [1707], matching LYSINE (Daicel p38).
+ Chiralpak WE [1707], matching LYSINE (Daicel p38).
+ Crownpak CR [1708], matching DL-TYROSINE (Daicel p35).
+ Crownpak CR [1772], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Crownpak CR [1804], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [1804], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [1804], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [1804], matching DL-VALINE (Daicel p33).
+ Crownpak CR [1852], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [1852], matching L-ORNITHINE (Daicel p38).
+ Chiralcel OD [1858], matching CARBINOXAMINE (Daicel p21).
+ Chiralcel OB [2021], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
+ Crownpak CR [2023], matching ALANINE (Daicel p33).
+ Chiralpak WH [2023], matching ALANINE (Daicel p33).
+ Chiralpak WM [2023], matching ALANINE (Daicel p33).
+ Crownpak CR [2161], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL (Daicel p32).
+ Chiralcel OB [2191], matching N,O-DIBENZOYL2-AMINOBUTANOL (Daicel p46).
+ Chiralcel OJ [2241], matching TRIADIMEFON (Daicel p22).
+ Chiralcel OD [3116], matching 2 BUTYL PHENYL CARBAMATE (Daicel p46).

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions (column) for molecule PINDOLOL....
+ Chiralcel OD [369], matching PRACTOLOL (Daicel p20).
+ Chiralcel OD [497], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OD [573], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [573], matching PERISOXAL (Daicel p20).
+ Chiralcel OD [817], matching CARTEOLOL (Daicel p20).
+ Crownpak CR [1059], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [1059], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralcel OB [1304], matching 1 PHENYL 1,3 PROPANEDIOL (Daicel p44).
+ Chiralcel CA-1 [1425], matching MANDELAMIDE (Daicel p44).
+ Chiralcel OB [1474], matching 3 PHENYL BUTYL BENZOATE (Daicel p45).
+ Crownpak CR [1434], matching LYSINE (Daicel p38).
+ Chiralpak WM [1434], matching LYSINE (Daicel p38).
+ Chiralpak WE [1434], matching LYSINE (Daicel p38).
+ Chiralcel OB [1470], matching 1-PHENYL-1,2-ETHANEDIOL (Daicel p44).
+ Crownpak CR [1553], matching HISTIDINE (Daicel p38).
+ Chiralpak WH [1553], matching HISTIDINE (Daicel p38).
+ Chiralcel OD [1575], matching GUAIFENESINE (Daicel p44).
+ Crownpak CR [1578], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [1578], matching L-ORNITHINE (Daicel p38).
+ Crownpak CR [1660], matching L-GLUTAMINE (Daicel p36).
+ Crownpak CR [1803], matching THREONINE (Daicel p35).
+ Chiralpak WH [1803], matching THREONINE (Daicel p35).
+ Crownpak CR [1977], matching SERINE (Daicel p35).
+ Chiralpak WH [1977], matching SERINE (Daicel p35).

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions (column) for molecule METOPROLOL....
+ Chiralcel OD [107], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OD [237], matching PROPRANOLOL (Daicel p20).
+ Chiralcel OD [621], matching PERISOXAL (Daicel p20).
+ Chiralcel OF [621], matching PERISOXAL (Daicel p20).
+ Chiralcel OD [681], matching PRACTOLOL (Daicel p20).
+ Chiralcel OD [1064], matching BENZOIN (Daicel p44).
+ Chiralcel OF [1064], matching BENZOIN (Daicel p44).
+ Chiralpak OP [1064], matching BENZOIN (Daicel p44).
+ Chiralpak WH [1115], matching N-CBZ-PHENYLALANINE (Daicel p40).
+ Chiralcel OD [1121], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [1121], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [1121], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel CA-1 [1121], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak WH [1128], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1128], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1134], matching BENZYL PHENYL CARBINOL (Daicel p46).
+ Chiralcel OJ [1168], matching KETOPROFEN (Daicel p27).
+ Chiralcel OJ [1194], matching FENOPROFEN (Daicel p27).
+ Chiralcel OJ [1210], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p44).
+ Chiralcel OB [1210], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p44).
+ Chiralcel OK [1210], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [1210], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p44).
+ Chiralcel OC [1210], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p44).
+ Chiralcel OJ [1210], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [1210], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralpak WH [1295], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1295], matching N-CBZ-VALINE (Daicel p40).
+ Crownpak CR [1350], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralcel OK [1393], matching METHYL MANDELATE (Daicel p44).
+ Crownpak CR [1429], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [1429], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [1513], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [1513], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OB [1593], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Crownpak CR [1610], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [1610], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [1610], matching D-PHENYLALANINE (Daicel p34).
+ Crownpak CR [1646], matching LYSINE (Daicel p38).
+ Chiralpak WM [1646], matching LYSINE (Daicel p38).
+ Chiralpak WE [1646], matching LYSINE (Daicel p38).
+ Crownpak CR [1659], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID (Daicel p35).
+ Chiralcel OB [1663], matching 1-PHENYLETHANOL (Daicel p42).
+ Chiralpak WM [1716], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [1716], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1716], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [1716], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [1722], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Crownpak CR [1754], matching PHENYLGLYCINE (Daicel p34).
+ Crownpak CR [1789], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [1789], matching L-ORNITHINE (Daicel p38).

+ Crownpak CR [1803], matching 2-AMINO 1-HEXANOL {Daicel p42}.
+ Chiralpak WE [1832], matching DL LEUCINE {Daicel p33}.
+ Chiralpak WM [1832], matching DL LEUCINE {Daicel p33}.
+ Crownpak CR [1835], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1835], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1835], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1835], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1836], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OB [1878], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1878], matching 1,3-DIACETOXY BUTANE {Daicel p45}.
+ Chiralcel OJ [1902], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1902], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1910], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1927], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [1941], matching CARTEOLOL {Daicel p20}.
+ Crownpak CR [1945], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [1977], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1977], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1977], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1977], matching DL-VALINE {Daicel p33}.
+ Chiralcel OD [2061], matching CARBINOXAMINE {Daicel p21}.
+ Crownpak CR [2071], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [2079], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Crownpak CR [2120], matching THREONINE {Daicel p35}.
+ Chiralpak WH [2120], matching THREONINE {Daicel p35}.
+ Chiralcel OB [2153], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Crownpak CR [2196], matching ALANINE {Daicel p33}.
+ Chiralpak WH [2196], matching ALANINE {Daicel p33}.
+ Chiralpak WM [2196], matching ALANINE {Daicel p33}.
+ Crownpak CR [2209], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [2209], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [2209], matching ASPARTIC ACID {Daicel p37}.
+ Chiralcel OB [2268], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [2327], matching SERINE {Daicel p35}.
+ Chiralpak WH [2327], matching SERINE {Daicel p35}.
+ Crownpak CR [2408], matching ERYTHRO-2-AMINO-1,2 DIPHENYLETHANOL {Daicel p32}.
+ Crownpak CR [3041], matching DL-TYROSINE {Daicel p35}.
+ Chiralcel OD [3320], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule ALPRENOLOL....
+ Chiralcel OD [67], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [107], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [333], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [333], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [497], matching PINDOLOL {Daicel p20}.
+ Chiralcel OD [652], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OD [862], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [862], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [862], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [869], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OD [892], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [970], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [970], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [970], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [970], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [970], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [970], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [970], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1034], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1034], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1034], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1034], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Crownpak CR [1075], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [1110], matching N-CBZ-NOR LEUCINE {Daicel p40}.
+ Chiralcel OD [1110], matching N-CBZ-NOR LEUCINE {Daicel p40}.
+ Chiralcel OB [1112], matching 3-PHENYL BUTYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [1129], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OK [1138], matching METHYL MANDELATE {Daicel p44}.
+ Chiralpak WH [1141], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OJ [1190], matching KETOPROFEN {Daicel p27}.
+ Crownpak CR [1310], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1310], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1310], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1321], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1321], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel CA-1 [1327], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [1338], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OH [1414], matching 1-PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [1440], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [1463], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [1477], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1477], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1477], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1477], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1493], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Chiralpak WH [1540], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1540], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralpak WH [1541], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OB [1622], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OJ [1646], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1646], matching CYCLOPENTOLATE {Daicel p21}.
+ Crownpak CR [1648], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Chiralcel OD [1654], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1663], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [1666], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1666], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1666], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1677], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1677], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1680], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1680], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1680], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1680], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralcel OB [1693], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Crownpak CR [1697], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [1791], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [1811], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1811], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [1822], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1822], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1822], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1822], matching DL-VALINE {Daicel p33}.
+ Chiralcel OD [1876], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OB [2013], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [2041], matching ALANINE {Daicel p33}.
+ Chiralpak WH [2041], matching ALANINE {Daicel p33}.
+ Chiralpak WM [2041], matching ALANINE {Daicel p33}.
+ Chiralcel OB [2150], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Crownpak CR [2152], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OJ [2233], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OD [3134], matching 2 BUTYL PHENYL CARBAMATE {Daicel p46}.

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule PERISOXAL....
+ Chiralcel OD [333], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [357], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [573], matching PINDOLOL {Daicel p20}.
+ Chiralcel OD [621], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [751], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [751], matching BENZOIN {Daicel p44}.

+ Chiralpak OP [751], matching BENZOIN (Daicel p44).
+ Chiralcel OD [802], matching BENZYL PHENYL CARBINOL (Daicel p46).
+ Chiralcel OJ [981], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [981], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OK [981], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [981], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OC [981], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [981], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [981], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Crownpak CR [1051], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
+ Chiralpak WH [1085], matching N-CBZ-PHENYLALANINE (Daicel p40).
+ Chiralcel OK [1123], matching METHYL MANDELATE (Daicel p44).
+ Crownpak CR [1285], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [1285], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [1285], matching D-PHENYLALANINE (Daicel p34).
+ Chiralcel OJ [1307], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel OD [1307], matching CYCLOPENTOLATE (Daicel p21).
+ Chiralcel CA-1 [1315], matching MANDELAMIDE (Daicel p44).
+ Chiralpak WH [1315], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [1315], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OB [1317], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OD [1392], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [1392], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [1392], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel CA-1 [1392], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OB [1393], matching 1-PHENYLETHANOL (Daicel p42).
+ Crownpak CR [1406], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID (Daicel p35).
+ Crownpak CR [1416], matching 2-AMINO-2-PHENYLETHANOL (Daicel p32).
+ Chiralcel OB [1434], matching 1,3 BUTANEDIOL DIBENZOATE (Daicel p45).
+ Crownpak CR [1439], matching PHENYLGLYCINE (Daicel p34).
+ Chiralpak WH [1457], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [1457], matching N-CBZ-VALINE (Daicel p40).
+ Chiralpak WM [1464], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [1464], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1464], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [1464], matching L-METHIONINE (Daicel p36).
+ Chiralcel OJ [1470], matching FENOPROFEN (Daicel p27).
+ Chiralpak WH [1516], matching 2-AMINO-3-PHENYLPROPIONAMIDE (Daicel p41).
+ Chiralcel OB [1521], matching 1,4 PENTYL DIBENZOATE (Daicel p45).
+ Crownpak CR [1561], matching 2-AMINO-1-HEXANOL (Daicel p32).
+ Chiralpak WE [1590], matching DL-LEUCINE (Daicel p33).
+ Chiralpak WM [1590], matching DL-LEUCINE (Daicel p33).
+ Crownpak CR [1593], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WH [1593], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WM [1593], matching L-ISOLEUCINE (Daicel p34).
+ Chiralpak WE [1593], matching L-ISOLEUCINE (Daicel p34).
+ Chiralcel OB [1602], matching 2-PHENYL-PROPAN-1-OL (Daicel p42).
+ Chiralcel OB [1605], matching 1,3-DIACETOXY-BUTANE (Daicel p45).
+ Chiralcel OD [1634], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralcel OB [1647], matching 2-ACETOXY-3-PHENYLPROPANE (Daicel p45).
+ Chiralcel OJ [1665], matching KETOPROFEN (Daicel p27).
+ Chiralpak WH [1676], matching N-CBZ-ALANINE (Daicel p40).
+ Chiralcel OD [1676], matching N-CBZ-ALANINE (Daicel p40).
+ Crownpak CR [1699], matching DL-TYROSINE (Daicel p35).
+ Crownpak CR [1704], matching 2-AMINO-1-PENTANOL (Daicel p32).
+ Crownpak CR [1735], matching DL-VALINE (Daicel p33).
+ Chiralpak WH [1735], matching DL-VALINE (Daicel p33).
+ Chiralpak WM [1735], matching DL-VALINE (Daicel p33).
+ Chiralpak WE [1735], matching DL-VALINE (Daicel p33).
+ Crownpak CR [1745], matching LYSINE (Daicel p38).
+ Chiralpak WM [1745], matching LYSINE (Daicel p38).
+ Chiralpak WE [1745], matching LYSINE (Daicel p38).
+ Chiralcel OD [1785], matching CARBINOXAMINE (Daicel p21).
+ Crownpak CR [1889], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [1889], matching L-ORNITHINE (Daicel p38).
+ Crownpak CR [1954], matching ALANINE (Daicel p33).
+ Chiralpak WH [1954], matching ALANINE (Daicel p33).
+ Chiralpak WM [1954], matching ALANINE (Daicel p33).
+ Chiralcel OB [1992], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
+ Crownpak CR [2128], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL (Daicel p32).
+ Chiralcel OB [2151], matching N,O-DIBENZOYL2-AMINOBUTANOL (Daicel p46).
+ Chiralcel OJ [2226], matching TRIADIMEFON (Daicel p22).
+ Chiralcel OD [3044], matching 2 BUTYL PHENYL CARBAMATE (Daicel p46).

****

*** Testing for suggestions for column Chiralcel OD ....
= Ordered column suggestions [column] for molecule N-CBZ-ALANINE....
+ Chiralpak WH [219], matching N-CBZ-VALINE (Daicel p40).
+ Chiralcel OD [219], matching N-CBZ-VALINE (Daicel p40).
+ Chiralpak WH [361], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralcel OD [361], matching N-CBZ-NOR-LEUCINE (Daicel p40).
+ Chiralpak WH [587], matching N-CBZ-PHENYLALANINE (Daicel p40).
+ Chiralcel OJ [650], matching FENOPROFEN (Daicel p27).
+ Chiralcel OJ [657], matching KETOPROFEN (Daicel p27).
+ Chiralcel OB [904], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OC [904], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [904], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [904], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OJ [904], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OB [904], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OK [904], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
+ Chiralcel OD [1014], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OG [1014], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralpak OT [1014], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel CA-1 [1014], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE (Daicel p43).
+ Chiralcel OB [1048], matching 1,3 BUTANEDIOL DIBENZOATE (Daicel p45).
+ Chiralcel OB [1054], matching 1,4 PENTYL DIBENZOATE (Daicel p45).
+ Chiralcel OB [1088], matching 1,3-DIACETOXY-BUTANE (Daicel p45).
+ Chiralcel OD [1108], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
+ Chiralcel OB [1134], matching 2-ACETOXY-3-PHENYLPROPANE (Daicel p45).
+ Crownpak CR [1272], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralpak WH [1272], matching D-(+)-TRYPTOPHAN (Daicel p38).
+ Chiralcel OB [1298], matching 1,5 HEXYL DIBENZOATE (Daicel p45).
+ Crownpak CR [1329], matching LYSINE (Daicel p38).
+ Chiralpak WM [1329], matching LYSINE (Daicel p38).
+ Chiralpak WE [1329], matching LYSINE (Daicel p38).
+ Chiralcel OB [1348], matching 1-PHENYLETHANOL (Daicel p42).
+ Chiralcel OB [1424], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
+ Crownpak CR [1428], matching ALANINE (Daicel p33).
+ Chiralpak WH [1428], matching ALANINE (Daicel p33).
+ Chiralpak WM [1428], matching ALANINE (Daicel p33).
+ Chiralcel OB [1464], matching 1 METHYL 3 PHENYL PROPYL BENZOATE (Daicel p45).
+ Crownpak CR [1473], matching L-ORNITHINE (Daicel p38).
+ Chiralpak WM [1473], matching L-ORNITHINE (Daicel p38).
+ Chiralcel OB [1483], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
+ Crownpak CR [1503], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WH [1503], matching D-PHENYLALANINE (Daicel p34).
+ Chiralpak WM [1503], matching D-PHENYLALANINE (Daicel p34).
+ Chiralcel OD [1514], matching METOPROLOL (Daicel p20).
+ Chiralcel OD [1540], matching ALPRENOLOL (Daicel p20).
+ Chiralcel OD [1577], matching BENZOIN (Daicel p44).
+ Chiralcel OF [1577], matching BENZOIN (Daicel p44).
+ Chiralpak OP [1577], matching BENZOIN (Daicel p44).
+ Chiralcel OB [1605], matching ALPHA METHYL PHENETHYL BENZOATE (Daicel p45).
+ Chiralpak WM [1610], matching L-METHIONINE (Daicel p36).
+ Crownpak CR [1610], matching L-METHIONINE (Daicel p36).
+ Chiralpak WH [1610], matching L-METHIONINE (Daicel p36).
+ Chiralpak WE [1610], matching L-METHIONINE (Daicel p36).
+ Chiralcel OD [1646], matching BENZYL PHENYL CARBINOL (Daicel p46).

```
 + Crownpak CR [1646], matching DL-VALINE (Daicel p33).
 + Chiralpak WH [1646], matching DL-VALINE (Daicel p33).
 + Chiralpak WM [1646], matching DL-VALINE (Daicel p33).
 + Chiralpak WE [1646], matching DL-VALINE (Daicel p33).
 + Crownpak CR [1647], matching PHENYLGLYCINE (Daicel p34).
 + Chiralcel OK [1648], matching METHYL MANDELATE (Daicel p44).
 + Chiralcel OD [1666], matching PROPRANOLOL (Daicel p20).
 + Crownpak CR [1668], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
         (Daicel p35).
 + Chiralcel OB [1711], matching METHYL-3-BENZOYLOXYPROPANOATE (Daicel p45).
 + Chiralpak WE [1725], matching DL-LEUCINE (Daicel p33).
 + Chiralpak WM [1725], matching DL-LEUCINE (Daicel p33).
 + Crownpak CR [1729], matching L-ISOLEUCINE (Daicel p34).
 + Chiralpak WH [1729], matching L-ISOLEUCINE (Daicel p34).
 + Chiralpak WM [1729], matching L-ISOLEUCINE (Daicel p34).
 + Chiralpak WE [1729], matching L-ISOLEUCINE (Daicel p34).
 + Chiralpak OT [1749], matching 2 ETHYL PHENYL BENZOATE (Daicel p46).
 + Crownpak CR [1811], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
 + Chiralcel OJ [1865], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
 + Chiralcel OK [1865], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
 + Chiralcel OD [1931], matching PERISOXAL (Daicel p20).
 + Chiralcel OF [1931], matching PERISOXAL (Daicel p20).
 + Chiralcel OB [2116], matching PENTYNYL 3 BENZOATE (Daicel p45).
 + Chiralcel OB [2271], matching BUTYNYL 3 BENZOATE (Daicel p45).
 + Chiralcel OB [2523], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
(Daicel p29).
 + Chiralcel OB [2666], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE (Daicel p29).

************************************************************
*** Testing for suggestions for column Chiralcel OD ....
 - Ordered column suggestions [column] for molecule N-CBZ-NOR-LEUCINE....
 + Chiralpak WH [70], matching N-CBZ-VALINE (Daicel p40).
 + Chiralcel OD [70], matching N-CBZ-VALINE (Daicel p40).
 + Chiralpak WH [191], matching N-CBZ-PHENYLALANINE (Daicel p40).
 + Chiralpak WH [361], matching N-CBZ-ALANINE (Daicel p40).
 + Chiralcel OD [361], matching N-CBZ-ALANINE (Daicel p40).
 + Chiralcel OJ [1011], matching FENOPROFEN (Daicel p27).
 + Chiralcel OJ [1018], matching KETOPROFEN (Daicel p27).
 + Chiralcel OD [1110], matching ALPRENOLOL (Daicel p20).
 + Chiralcel OD [1128], matching METOPROLOL (Daicel p20).
 + Chiralcel OB [1265], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OC [1265], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OJ [1265], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OD [1265], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OJ [1265], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OB [1265], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OK [1265], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OD [1280], matching PROPRANOLOL (Daicel p20).
 + Chiralcel OD [1315], matching PERISOXAL (Daicel p20).
 + Chiralcel OF [1315], matching PERISOXAL (Daicel p20).
 + Chiralcel OD [1323], matching BENZOIN (Daicel p44).
 + Chiralcel OF [1323], matching BENZOIN (Daicel p44).
 + Chiralpak OP [1323], matching BENZOIN (Daicel p44).
 + Chiralcel OD [1375], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         (Daicel p43).
 + Chiralcel OG [1375], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         (Daicel p43).
 + Chiralpak OT [1375], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         (Daicel p43).
 + Chiralcel CA-1 [1375], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         (Daicel p43).
 + Chiralcel OD [1393], matching BENZYL PHENYL CARBINOL (Daicel p46).
 + Chiralcel OB [1409], matching 1,3 BUTANEDIOL DIBENZOATE (Daicel p45).
 + Chiralcel OB [1415], matching 1,4 PENTYL DIBENZOATE (Daicel p45).
 + Crownpak CR [1428], matching L-ISOLEUCINE (Daicel p34).
 + Chiralpak WH [1428], matching L-ISOLEUCINE (Daicel p34).
 + Chiralpak WM [1428], matching L-ISOLEUCINE (Daicel p34).
 + Chiralpak WE [1428], matching L-ISOLEUCINE (Daicel p34).
 + Chiralpak WE [1431], matching DL-LEUCINE (Daicel p33).
 + Chiralpak WM [1431], matching DL-LEUCINE (Daicel p33).
 + Chiralcel OB [1449], matching 1,3 DIACETOXY-BUTANE (Daicel p45).
 + Chiralcel OD [1469], matching 2-PHENYLPROPIONIC ACID (Daicel p27).
 + Chiralcel OB [1495], matching 2-ACETOXY-3-PHENYLPROPANE (Daicel p45).
 + Crownpak CR [1559], matching 4-AMINO-3 (P CHLOROPHENYL)BUTYRIC ACID
         (Daicel p35).
 + Crownpak CR [1570], matching DL-VALINE (Daicel p33).
 + Chiralpak WH [1570], matching DL-VALINE (Daicel p33).
 + Chiralpak WM [1570], matching DL-VALINE (Daicel p33).
 + Chiralpak WE [1570], matching DL-VALINE (Daicel p33).
 + Crownpak CR [1616], matching PHENYLGLYCINE (Daicel p34).
 + Chiralpak WM [1622], matching L-METHIONINE (Daicel p36).
 + Crownpak CR [1622], matching L-METHIONINE (Daicel p36).
 + Chiralpak WH [1622], matching L-METHIONINE (Daicel p36).
 + Chiralpak WE [1622], matching L-METHIONINE (Daicel p36).
 + Chiralcel OB [1633], matching 1-PHENYL-PROPAN-1-OL (Daicel p42).
 + Crownpak CR [1653], matching D-PHENYLALANINE (Daicel p34).
 + Chiralpak WH [1653], matching D-PHENYLALANINE (Daicel p34).
 + Chiralpak WM [1653], matching D-PHENYLALANINE (Daicel p34).
 + Chiralcel OB [1659], matching 1,5 HEXYL DIBENZOATE (Daicel p45).
 + Crownpak CR [1663], matching 2-AMINO-1-PHENYLETHANOL (Daicel p32).
 + Crownpak CR [1690], matching LYSINE (Daicel p38).
 + Chiralpak WM [1690], matching LYSINE (Daicel p38).
 + Chiralpak WE [1690], matching LYSINE (Daicel p38).
 + Chiralcel OK [1703], matching METHYL MANDELATE (Daicel p44).
 + Chiralcel OB [1709], matching 1-PHENYLETHANOL (Daicel p42).
 + Crownpak CR [1789], matching ALANINE (Daicel p33).
 + Chiralpak WH [1789], matching ALANINE (Daicel p33).
 + Chiralpak WM [1789], matching ALANINE (Daicel p33).
 + Chiralcel OB [1825], matching 1 METHYL 3 PHENYL PROPYL BENZOATE (Daicel p45).
 + Crownpak CR [1834], matching L-ORNITHINE (Daicel p38).
 + Chiralpak WM [1834], matching L-ORNITHINE (Daicel p38).
 + Chiralcel OB [1844], matching N-(PHENYLETHYL)ACETAMIDE (Daicel p46).
 + Chiralcel OB [1966], matching ALPHA METHYL PHENETHYL BENZOATE (Daicel p45).
 + Chiralcel OB [2072], matching METHYL-3-BENZOYLOXYPROPANOATE (Daicel p45).
 + Chiralpak OT [2110], matching 2 ETHYL PHENYL BENZOATE (Daicel p46).
 + Chiralcel OJ [2226], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
 + Chiralcel OK [2226], matching METHYL 2 PHENOXYPROPIONATE (Daicel p29).
 + Chiralcel OB [2477], matching PENTYNYL 3 BENZOATE (Daicel p45).
 + Chiralcel OB [2632], matching BUTYNYL 3 BENZOATE (Daicel p45).
 + Chiralcel OB [2884], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
(Daicel p29).
 + Chiralcel OB [3027], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE (Daicel p29).

************************************************************
*** Testing for suggestions for column Chiralcel OD ....
 = Ordered column suggestions [column] for molecule N-CBZ-VALINE....
 + Chiralpak WH [70], matching N-CBZ-NOR-LEUCINE (Daicel p40).
 + Chiralcel OD [70], matching N-CBZ-NOR-LEUCINE (Daicel p40).
 + Chiralpak WH [219], matching N-CBZ-ALANINE (Daicel p40).
 + Chiralcel OD [219], matching N-CBZ-ALANINE (Daicel p40).
 + Chiralpak WH [368], matching N-CBZ-PHENYLALANINE (Daicel p40).
 + Chiralpak WH [865], matching N-CBZ-ASPARAGINE (Daicel p40).
 + Chiralcel OD [865], matching N-CBZ-ASPARAGINE (Daicel p40).
 + Chiralcel OJ [869], matching FENOPROFEN (Daicel p27).
 + Chiralcel OJ [876], matching KETOPROFEN (Daicel p27).
 + Chiralcel OB [1123], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OC [1123], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OJ [1123], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OD [1123], matching 1-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OJ [1123], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OB [1123], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OK [1123], matching 2-(1-HYDROXYETHYL)NAPHTHALENE (Daicel p43).
 + Chiralcel OD [1233], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         (Daicel p43).
 + Chiralcel OG [1233], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
         (Daicel p43).
```

```
+ Chiralpak OT [1233], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
        {Daicel p43}.
+ Chiralcel CA-1 [1233], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
        {Daicel p43}.
+ Chiralcel OR [1267], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OR [1273], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [1295], matching METOPROLOL {Daicel p20}.
+ Chiralcel OR [1307], matching 1.3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [1321], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1326], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OR [1353], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [1358], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1358], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1358], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [1427], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Crownpak CR [1428], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1428], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1428], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1428], matching DL-VALINE {Daicel p33}.
+ Chiralcel OD [1447], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1457], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1457], matching PERISOXAL {Daicel p20}.
+ Chiralcel OR [1491], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OR [1517], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1547], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1547], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1547], matching LYSINE {Daicel p38}.
+ Chiralcel OR [1567], matching 1-PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [1570], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1570], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1570], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1570], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1573], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1573], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WH [1591], matching N-ACETYL TRYPTOPHAN {Daicel p39}.
+ Crownpak CR [1646], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1646], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1646], matching ALANINE {Daicel p33}.
+ Crownpak CR [1651], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralcel OR [1683], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralpak WM [1689], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1689], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1689], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1689], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1692], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1692], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OK [1692], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [1697], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [1701], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
        {Daicel p35}.
+ Chiralcel OR [1702], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1722], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1722], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1722], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OR [1823], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OR [1930], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [1968], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [2084], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [2084], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OR [2335], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OR [2490], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OR [2742], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OR [2885], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY] PROPIONATE {Daicel p29}.

**************************************************************
*** Testing for suggestions for column Chiralcel OD ....
- Ordered column suggestions [column] for molecule N-CBZ-ASPARAGINE....
+ Chiralpak WH [727], matching N-ACETYL TRYPTOPHAN {Daicel p39}.
+ Chiralpak WH [865], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [865], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [963], matching N-ACETYL-L PHENYLALANINE {Daicel p39}.
+ Chiralcel OD [1205], matching GUAIFENESINE {Daicel p44}.
+ Chiralpak WH [1331], matching N-ACTEYL VALINE {Daicel p39}.
+ Crownpak CR [1428], matching ASPARAGINE {Daicel p37}.
+ Chiralcel CA-1 [1512], matching MANDELAMIDE {Daicel p44}.
+ Crownpak CR [1609], matching L-GLUTAMINE {Daicel p36}.
+ Chiralcel OR [1694], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralpak WH [1706], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [1706], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [1736], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [1736], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [1736], matching ASPARTIC ACID {Daicel p37}.
+ Chiralcel OR [1745], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [1773], matching THREONINE {Daicel p35}.
+ Chiralpak WH [1773], matching THREONINE {Daicel p35}.
+ Crownpak CR [1787], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1787], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [1825], matching SERINE {Daicel p35}.
+ Chiralpak WH [1825], matching SERINE {Daicel p35}.
+ Crownpak CR [1842], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OD [1925], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OJ [1938], matching TRIADIMEFON {Daicel p22}.
+ Chiralpak WH [2149], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [2410], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [2410], matching D-(+)-TRYPTOPHAN {Daicel p38}.

**************************************************************
*** Testing for suggestions for column Chiralcel OD ....
- Ordered column suggestions [column] for molecule 2PHENOXYPROPIONIC ACID....
+ Chiralcel OD [334], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OJ [456], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [456], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OR [466], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [615], matching DL-TYROSINE {Daicel p35}.
+ Chiralcel OR [615], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OR [618], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OR [675], matching 1.3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OR [694], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OR [710], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OR [774], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OJ [789], matching METHYL2-PHENYLPROPIONATE {Daicel p28}.
+ Chiralpak WH [798], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [805], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Crownpak CR [818], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralpak OT [857], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Crownpak CR [879], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [929], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [929], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [958], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [958], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [976], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [976], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [976], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak OT [980], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [981], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OR [1006], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1006], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1006], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1006], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1006], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OR [1006], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1006], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Crownpak CR [1011], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [1032], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [1032], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [1032], matching ASPARTIC ACID {Daicel p37}.
```

+ Chiralcel OB [1077], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [1081], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [1090], matching 1-PHENYL-1,2 ETHANEDIOL {Daicel p44}.
+ Crownpak CR [1100], matching CITRULLINE {Daicel p37}.
+ Chiralcel OJ [1139], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OB [1173], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Crownpak CR [1203], matching ARGININE {Daicel p39}.
+ Chiralpak WE [1203], matching ARGININE {Daicel p39}.
+ Chiralpak WM [1203], matching ARGININE {Daicel p39}.
+ Chiralcel OB [1253], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel CA-1 [1283], matching MANDELAMIDE {Daicel p44}.
+ Chiralpak WH [1332], matching 2 AMINO 3 PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [1364], matching 2 AMINO 1 PHENYLETHANOL {Daicel p42}.
+ Chiralcel OD [1424], matching 9 (2 TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1424], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1424], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1424], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OJ [1432], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralcel OB [1593], matching 3 PHENYL BUTYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1972], matching PRACTOLOL {Daicel p20}.

****

*** Testing for suggestions for column Chiralpak WH ....
- Ordered column suggestions [column] for molecule MANDELIC ACID....
+ Chiralcel OB [133], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralcel OB [200], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel CA-1 [325], matching MANDELAMIDE {Daicel p44}.
+ Crownpak CR [344], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [355], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [355], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [355], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [355], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [406], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [443], matching LACTIC ACID {Daicel p27}.
+ Chiralcel OK [456], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [476], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [507], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [507], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [507], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [507], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [566], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [566], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [566], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [566], matching DL-VALINE {Daicel p33}.
+ Chiralcel OD [625], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [626], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [642], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [652], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [711], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [737], matching LYSINE {Daicel p38}.
+ Chiralpak WM [737], matching LYSINE {Daicel p38}.
+ Chiralpak WE [737], matching LYSINE {Daicel p38}.
+ Crownpak CR [753], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [753], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WH [754], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Chiralcel OB [757], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralpak WH [798], matching N-ACTEYL VALINE {Daicel p39}.
+ Crownpak CR [930], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [930], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [935], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OD [958], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [958], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Crownpak CR [1008], matching 1-PHENYLETHYLAMINE {Daicel p42}.
+ Crownpak CR [1072], matching SERINE {Daicel p35}.
+ Chiralpak WH [1072], matching SERINE {Daicel p35}.
+ Chiralpak OT [1148], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Crownpak CR [1239], matching THREONINE {Daicel p35}.
+ Chiralpak WH [1239], matching THREONINE {Daicel p35}.
+ Crownpak CR [1265], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [1265], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [1265], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [1280], matching CITRULLINE {Daicel p37}.
+ Crownpak CR [1462], matching ASPARAGINE {Daicel p37}.
+ Chiralcel OJ [1532], matching TRIADIMEFON {Daicel p22}.
+ Crownpak CR [1643], matching L-GLUTAMINE {Daicel p36}.
+ Chiralcel OD [1692], matching PRACTOLOL {Daicel p20}.
+ Chiralpak WH [1706], matching N CBZ ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1706], matching N CBZ ASPARAGINE {Daicel p40}.
+ Crownpak CR [1949], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [1949], matching D-(+)-TRYPTOPHAN {Daicel p38}.

****

*** Testing for suggestions for column Chiralpak WH ....
- Ordered column suggestions [column] for molecule D PHENYLALANINE....
+ Chiralpak WM [112], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [112], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [112], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [112], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [156], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WE [191], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [191], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [191], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [191], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [191], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [191], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [304], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [325], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [329], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [336], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [351], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID {Daicel p35}.
+ Crownpak CR [368], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [368], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [368], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [368], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [399], matching LYSINE {Daicel p38}.
+ Chiralpak WM [399], matching LYSINE {Daicel p38}.
+ Chiralpak WE [399], matching LYSINE {Daicel p38}.
+ Crownpak CR [544], matching 2 AMINO 1 PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [573], matching L ORNITHINE {Daicel p38}.
+ Chiralpak WM [573], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [587], matching ALANINE {Daicel p33}.
+ Chiralpak WH [587], matching ALANINE {Daicel p33}.
+ Chiralpak WM [587], matching ALANINE {Daicel p33}.
+ Chiralcel OB [617], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [695], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OK [776], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [824], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Crownpak CR [900], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [901], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [961], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [976], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [976], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralcel OB [977], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OJ [1012], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1012], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1012], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1012], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1012], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1012], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1012], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Crownpak CR [1020], matching ARGININE {Daicel p39}.
+ Chiralpak WE [1020], matching ARGININE {Daicel p39}.

+ Chiralpak WM [1020], matching ARGININE {Daicel p39}.
+ Chiralcel OJ [1064], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OD [1068], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [1071], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OD [1078], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1078], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1078], matching BENZOIN {Daicel p44}.
+ Crownpak CR [1122], matching CITRULLINE {Daicel p37}.
+ Chiralcel OB [1175], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [1263], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1285], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1285], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1310], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OB [1315], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1319], matching PROPRANOLOL {Daicel p20}.
+ Chiralpak OT [1330], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OD [1400], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1400], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1400], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1400], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OJ [1426], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1426], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralpak WH [1428], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralpak WH [1503], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1503], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OB [1518], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1618], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1618], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1640], matching METOPROLOL {Daicel p20}.
+ Chiralcel OB [1693], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1694], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [1701], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [1722], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1722], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OB [1847], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1997], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2401], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OD [2437], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2543], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Chiralpak WH ....
- Ordered column suggestions [column] for molecule D-(+)-TRYPTOPHAN....
+ Crownpak CR [631], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [631], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [643], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [651], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Chiralcel OD [668], matching GUALFENESINE {Daicel p44}.
+ Crownpak CR [707], matching THREONINE {Daicel p35}.
+ Chiralpak WH [707], matching THREONINE {Daicel p35}.
+ Crownpak CR [781], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [781], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [781], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [821], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [861], matching SERINE {Daicel p35}.
+ Chiralpak WH [861], matching SERINE {Daicel p35}.
+ Crownpak CR [983], matching ASPARAGINE {Daicel p37}.
+ Chiralcel OD [1059], matching PINDOLOL {Daicel p20}.
+ Chiralpak WH [1272], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1272], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OJ [1314], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1314], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1314], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1314], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1314], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1314], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1314], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1429], matching METOPROLOL {Daicel p20}.
+ Chiralpak WH [1582], matching LACTIC ACID {Daicel p27}.
+ Chiralpak WH [1742], matching N-ACTEYL VALINE {Daicel p39}.
+ Chiralcel OB [1775], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralpak WH [1904], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [1904], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [1929], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralpak WH [2009], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Chiralcel CA-1 [2163], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OJ [2334], matching TRIADIMEFON {Daicel p22}.
+ Chiralpak WH [2410], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [2410], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [3021], matching PRACTOLOL {Daicel p20}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule HISTIDINE....
+ Crownpak CR [171], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [254], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [306], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [306], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [348], matching LYSINE {Daicel p38}.
+ Chiralpak WM [348], matching LYSINE {Daicel p38}.
+ Chiralpak WE [348], matching LYSINE {Daicel p38}.
+ Crownpak CR [353], matching CITRULLINE {Daicel p37}.
+ Crownpak CR [360], matching ASPARAGINE {Daicel p37}.
+ Crownpak CR [401], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [401], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [401], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [434], matching THREONINE {Daicel p35}.
+ Chiralpak WH [434], matching THREONINE {Daicel p35}.
+ Crownpak CR [586], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [600], matching SERINE {Daicel p35}.
+ Chiralpak WH [600], matching SERINE {Daicel p35}.
+ Chiralpak WM [606], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [606], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [606], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [606], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [631], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [631], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Crownpak CR [727], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [767], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [767], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [767], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [767], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [891], matching ALANINE {Daicel p33}.
+ Chiralpak WH [891], matching ALANINE {Daicel p33}.
+ Chiralpak WM [891], matching ALANINE {Daicel p33}.
+ Crownpak CR [912], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralcel OD [929], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [929], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralpak WH [930], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [930], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [1070], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralpak WH [1118], matching N-ACTEYL VALINE {Daicel p39}.
+ Chiralcel OD [1131], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1156], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Crownpak CR [1212], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralcel CA-1 [1255], matching MANDELAMIDE {Daicel p44}.
+ Chiralpak WH [1258], matching LACTIC ACID {Daicel p27}.
+ Chiralcel OB [1272], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1336], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WH [1386], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Chiralcel OD [1553], matching PINDOLOL {Daicel p20}.
+ Chiralcel OB [1593], matching 1-PHENYLETHANOL {Daicel p42}.

+ Chiralcel OB [1609], matching 1 PHENYL PROPAN 1 OL {Daicel p42}.
+ Chiralcel OD [1618], matching PRACTOLOL {Daicel p20}.
+ Chiralpak OT [1654], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OJ [1674], matching TRIADIMEFON {Daicel p22}.
+ Chiralpak WH [1787], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1787], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1969], matching GUAIFENESINE {Daicel p44}.

****

*** Testing for suggestions for column Chiralpak WH ....
- Ordered column suggestions [column] for molecule L-METHIONINE....
+ Crownpak CR [11], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [112], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [112], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [112], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [126], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [126], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [126], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [126], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [126], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [126], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [143], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [223], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [223], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [223], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [223], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [243], matching LYSINE {Daicel p38}.
+ Chiralpak WM [243], matching LYSINE {Daicel p38}.
+ Chiralpak WE [243], matching LYSINE {Daicel p38}.
+ Crownpak CR [266], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [295], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [295], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WH [355], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [355], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [368], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [400], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [444], matching DL-TYROSINE {Daicel p35}.
+ Chiralpak WH [456], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [480], matching ALANINE {Daicel p33}.
+ Chiralpak WH [480], matching ALANINE {Daicel p33}.
+ Chiralpak WM [480], matching ALANINE {Daicel p33}.
+ Chiralcel OB [487], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [521], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralcel OD [535], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [577], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Crownpak CR [606], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [606], matching HISTIDINE {Daicel p38}.
+ Chiralcel OK [632], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [637], matching 1-PHENYLETHYLAMINE {Daicel p32}.
+ Chiralcel CA-1 [680], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [689], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [728], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [756], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [757], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [785], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [833], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [895], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Chiralcel OB [907], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p40}.
+ Chiralpak OT [1000], matching 1-BROMO ETHYLBENZENE {Daicel p46}.
+ Chiralcel OD [1179], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [1181], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1181], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1181], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1181], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1181], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1181], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1181], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralpak OT [1185], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OD [1187], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1187], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1187], matching BENZOIN {Daicel p44}.
+ Chiralcel OJ [1235], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [1286], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [1335], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OJ [1380], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1380], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1453], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OD [1464], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1464], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1477], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1487], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OB [1495], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1539], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralpak WH [1553], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1553], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1567], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1567], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1567], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1567], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OB [1587], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1596], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1610], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1610], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralpak WH [1650], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1650], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1716], matching METOPROLOL {Daicel p20}.
+ Chiralcel OB [1807], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1953], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2077], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [2103], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [2265], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2302], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Chiralpak WH ....
- Ordered column suggestions [column] for molecule DL-VALINE....
+ Crownpak CR [70], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [70], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [70], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [70], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [133], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Chiralpak WE [146], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [146], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [199], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [206], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [219], matching ALANINE {Daicel p33}.
+ Chiralpak WH [219], matching ALANINE {Daicel p33}.
+ Chiralpak WM [219], matching ALANINE {Daicel p33}.
+ Chiralpak WM [223], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [223], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [223], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [223], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [344], matching 2-AMINO-2-PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [359], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [359], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [368], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [368], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [368], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [514], matching THREONINE {Daicel p35}.
+ Chiralpak WH [514], matching THREONINE {Daicel p35}.
+ Crownpak CR [518], matching L-GLUTAMIC ACID {Daicel p36}.

+ Crownpak CR [547], matching LYSINE {Daicel p38}.
+ Chiralpak WM [547], matching LYSINE {Daicel p38}.
+ Chiralpak WE [547], matching LYSINE {Daicel p38}.
+ Chiralpak WH [566], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [566], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [623], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Crownpak CR [634], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [634], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [634], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [634], matching ASPARTIC ACID {Daicel p37}.
+ Chiralcel OB [711], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Chiralcel OD [739], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Crownpak CR [759], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [767], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [767], matching HISTIDINE {Daicel p38}.
+ Chiralcel OB [769], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel OK [855], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [865], matching ASPARAGINE {Daicel p37}.
+ Chiralcel OB [905], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel CA-1 [910], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [945], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [1020], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1051], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [1035], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1152], matching CITRULLINE {Daicel p37}.
+ Chiralcel OB [1291], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1310], matching METHYL-3 BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [1374], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralpak WH [1428], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1428], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OB [1429], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1429], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1429], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1429], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1429], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1429], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1429], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1444], matching METHYL 2-PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1444], matching METHYL 2-PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OD [1471], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [1473], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [1475], matching FENOPROFEN {Daicel p27}.
+ Chiralpak WH [1497], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1497], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1540], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1540], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1540], matching BENZOIN {Daicel p44}.
+ Chiralcel OB [1542], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OJ [1631], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [1646], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1646], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OB [1682], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1735], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1735], matching PERISOXAL {Daicel p20}.
+ Chiralpak WH [1795], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OB [1804], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1822], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1834], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1834], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1834], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1834], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OD [2008], matching METOPROLOL {Daicel p20}.
+ Chiralcel OB [2055], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [2061], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2208], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2245], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OD [2343], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2357], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule L ISOLEUCINE....
+ Chiralpak WE [1], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [70], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [70], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [70], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [70], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [126], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [126], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [126], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [126], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [133], matching 2-AMINO-1-HEXANOL {Daicel p32}.
+ Crownpak CR [141], matching PHENYLGLYCINE {Daicel p34}.
+ Crownpak CR [191], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [191], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [191], matching D-PHENYLALANINE {Daicel p34}.
+ Crownpak CR [206], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [286], matching 2-AMINO-2-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [360], matching LYSINE {Daicel p38}.
+ Chiralpak WM [360], matching LYSINE {Daicel p38}.
+ Chiralpak WE [360], matching LYSINE {Daicel p38}.
+ Crownpak CR [361], matching ALANINE {Daicel p33}.
+ Chiralpak WH [361], matching ALANINE {Daicel p33}.
+ Chiralpak WM [361], matching ALANINE {Daicel p33}.
+ Crownpak CR [464], matching L ORNITHINE {Daicel p38}.
+ Chiralpak WM [364], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [400], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralpak WH [507], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [507], matching MANDELIC ACID {Daicel p27}.
+ Chiralpak WH [535], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [575], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [601], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [636], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OB [652], matching 1 PHENYL 1,2-ETHANEDIOL {Daicel p44}.
+ Chiralcel OD [681], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [710], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralcel OK [796], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [807], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [836], matching L-GLUTAMINE {Daicel p36}.
+ Chiralcel CA-1 [852], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [874], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OB [886], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [962], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1036], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1048], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Chiralcel OB [1287], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1287], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1287], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1287], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1287], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1287], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1287], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1294], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralpak OT [1315], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [1333], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OB [1365], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1377], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1377], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1377], matching BENZOIN {Daicel p44}.

+ Chiralpak WH [1428], matching N-CBZ NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1428], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OB [1452], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [1461], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [1489], matching KETOPROFEN {Daicel p27}.
+ Chiralpak WH [1497], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1497], matching N CBZ-VALINE {Daicel p40}.
+ Chiralcel OJ [1547], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1547], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1548], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OD [1593], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1593], matching PERISOXAL {Daicel p20}.
+ Chiralcel OB [1616], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralpak WH [1618], matching N CBZ PHENYLALANINE {Daicel p40}.
+ Chiralcel OD [1662], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1680], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1692], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1692], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1692], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1692], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak WH [1729], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1729], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1866], matching METOPROLOL {Daicel p20}.
+ Chiralcel OB [1919], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2065], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2215], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2229], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OD [2339], matching CARTEOLOL {Daicel p20}.
+ Chiralcel OB [2418], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule ASPARTIC ACID....
+ Crownpak CR [92], matching THREONINE {Daicel p35}.
+ Chiralpak WH [92], matching THREONINE {Daicel p35}.
+ Crownpak CR [153], matching L GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [162], matching SERINE {Daicel p35}.
+ Chiralpak WH [162], matching SERINE {Daicel p35}.
+ Crownpak CR [241], matching ASPARAGINE {Daicel p37}.
+ Crownpak CR [386], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [401], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [401], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [529], matching GLYCERIC ACID {Daicel p27}.
+ Crownpak CR [592], matching ALANINE {Daicel p33}.
+ Chiralpak WH [592], matching ALANINE {Daicel p33}.
+ Chiralpak WM [592], matching ALANINE {Daicel p33}.
+ Crownpak CR [628], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [634], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [634], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [634], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [634], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [781], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [781], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Crownpak CR [786], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [873], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Chiralpak WH [942], matching LACTIC ACID {Daicel p27}.
+ Chiralpak WH [999], matching N-ACTEYL VALINE {Daicel p39}.
+ Chiralcel OD [1032], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [1032], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Crownpak CR [1134], matching 2 AMINO BUTANE {Daicel p32}.
+ Chiralpak WH [1265], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [1265], matching MANDELIC ACID {Daicel p27}.
+ Chiralpak WH [1267], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Chiralpak WH [1668], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1668], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OJ [1848], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OD [2051], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OD [2209], matching METOPROLOL {Daicel p20}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule LACTIC ACID....
+ Crownpak CR [344], matching ALANINE {Daicel p33}.
+ Chiralpak WH [344], matching ALANINE {Daicel p33}.
+ Chiralpak WM [344], matching ALANINE {Daicel p33}.
+ Chiralpak WH [443], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [443], matching MANDELIC ACID {Daicel p27}.
+ Chiralpak WH [500], matching GLYCERIC ACID {Daicel p27}.
+ Crownpak CR [733], matching SERINE {Daicel p35}.
+ Chiralpak WH [733], matching SERINE {Daicel p35}.
+ Chiralcel OD [798], matching 2PHENOXYPROPIONIC ACID {Daicel p29}.
+ Chiralcel OF [798], matching 2PHENOXYPROPIONIC ACID {Daicel p26}.
+ Chiralpak WH [818], matching N-ACTEYL VALINE {Daicel p39}.
+ Crownpak CR [838], matching THREONINE {Daicel p35}.
+ Chiralpak WH [838], matching THREONINE {Daicel p35}.
+ Crownpak CR [926], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [926], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [926], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [1023], matching 2 AMINO BUTANE {Daicel p32}.
+ Crownpak CR [1032], matching L-GLUTAMIC ACID {Daicel p36}.
+ Crownpak CR [1123], matching ASPARAGINE {Daicel p37}.
+ Chiralpak WH [1186], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Crownpak CR [1224], matching L-GLUTAMINE {Daicel p36}.
+ Chiralcel OJ [1253], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1253], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Crownpak CR [1258], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1258], matching HISTIDINE {Daicel p38}.
+ Chiralcel OB [1529], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OJ [1553], matching ETHYL 2-OXYPARAHYDROXYPHENYLPROPANOATE {Daicel p29}.
+ Crownpak CR [1611], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [1611], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Crownpak CR [1638], matching CITRULLINE {Daicel p37}.
+ Chiralcel OJ [1639], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [1764], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OJ [1986], matching TIAPROFENIC ACID ME ESTER {Daicel p28}.
+ Chiralcel OB [2004], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OJ [2040], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OD [2106], matching PRACTOLOL {Daicel p20}.
+ Chiralpak WH [2149], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [2149], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OB [2150], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2300], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule GLYCERIC ACID....
+ Crownpak CR [344], matching SERINE {Daicel p35}.
+ Chiralpak WH [344], matching SERINE {Daicel p35}.
+ Crownpak CR [439], matching THREONINE {Daicel p35}.
+ Chiralpak WH [439], matching THREONINE {Daicel p35}.
+ Chiralpak WH [500], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [529], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [529], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [529], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [733], matching ASPARAGINE {Daicel p37}.
+ Crownpak CR [881], matching ALANINE {Daicel p33}.
+ Chiralpak WH [881], matching ALANINE {Daicel p33}.
+ Chiralpak WM [881], matching ALANINE {Daicel p33}.
+ Chiralcel OD [1166], matching GUAIFENESINE {Daicel p44}.
+ Chiralpak WH [1453], matching N-ACETYL TRYPTOPHAN {Daicel p39}.

+ Crownpak CR [1523], matching 2 AMINO BUTANE {Daicel p32}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule WARFARINE....
+ Chiralcel OJ [802], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [802], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OB [1060], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralpak OT [1073], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OB [1158], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak WH [1188], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OB [1237], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OB [1329], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [1377], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [1440], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OJ [1485], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1485], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralpak WH [1537], matching N-AMIDE-L-PHENYLALANINE {Daicel p41}.
+ Chiralcel OB [1595], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1736], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1865], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OB [1882], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [2032], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule N-AMIDE-L-PHENYLALANINE....
+ Chiralpak WH [276], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Crownpak CR [815], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralpak WH [931], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Chiralcel OB [1098], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [1292], matching CITRULLINE {Daicel p37}.
+ Chiralcel OB [1535], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralpak WH [1537], matching WARFARINE {Daicel p39}.
+ Chiralcel OJ [1740], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1740], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OJ [1845], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1845], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [2069], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [2076], matching PENTYNYL 3 BENZOATE {Daicel p45}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule N-ACETYL TRYPTOPHAN....
+ Chiralpak WH [727], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [727], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralpak WH [1439], matching GLYCERIC ACID {Daicel p27}.
+ Chiralpak WH [1591], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1591], matching N-CBZ-VALINE {Daicel p40}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule N-ACETYL VALINE....
+ Chiralpak WH [368], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Chiralcel CA-1 [447], matching MANDELAMIDE {Daicel p44}.
+ Chiralcel OB [732], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Crownpak CR [759], matching ASPARAGINE {Daicel p37}.
+ Chiralpak WH [818], matching LACTIC ACID {Daicel p27}.
+ Chiralpak WH [823], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [823], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [830], matching 1-PHENYL-1,2-ETHANEDIOL {Daicel p44}.
+ Crownpak CR [932], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [999], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [999], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [999], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [1027], matching THREONINE {Daicel p35}.
+ Chiralpak WH [1027], matching THREONINE {Daicel p35}.
+ Crownpak CR [1118], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1118], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [1125], matching SERINE {Daicel p35}.
+ Chiralpak WH [1125], matching SERINE {Daicel p35}.
+ Crownpak CR [1174], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralpak WH [1331], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1331], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [1371], matching GUAIFENESINE {Daicel p44}.
+ Chiralcel OJ [1402], matching TRIADIMEFON {Daicel p22}.
+ Chiralcel OD [1684], matching PRACTOLOL {Daicel p20}.
+ Crownpak CR [1742], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [1742], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralcel OB [1750], matching N,O-DIBENZOYL2-AMINOBUTANOL {Daicel p46}.
+ Chiralcel OB [1878], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule N-ACETYL-L PHENYLALANINE....
+ Chiralpak WH [276], matching N-AMIDE-L-PHENYLALANINE {Daicel p41}.
+ Chiralpak WH [368], matching N-ACTEYL VALINE {Daicel p39}.
+ Chiralcel CA-1 [487], matching MANDELAMIDE {Daicel p44}.
+ Chiralpak WH [489], matching 2-AMINO-3-PHENYLPROPIONAMIDE {Daicel p41}.
+ Crownpak CR [524], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralcel OB [742], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralpak WH [754], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [754], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OB [793], matching 1-PHENYL 1,2 ETHANEDIOL {Daicel p44}.
+ Chiralcel OB [866], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralpak WH [963], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [963], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Crownpak CR [1027], matching ASPARAGINE {Daicel p37}.
+ Chiralcel OD [1035], matching GUAIFENESINE {Daicel p44}.
+ Chiralcel OJ [1088], matching TRIADIMEFON {Daicel p22}.
+ Chiralpak WH [1186], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [1208], matching L-GLUTAMINE {Daicel p36}.
+ Crownpak CR [1336], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [1336], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [1336], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [1373], matching THREONINE {Daicel p35}.
+ Chiralpak WH [1373], matching THREONINE {Daicel p35}.
+ Crownpak CR [1386], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1386], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [1424], matching SERINE {Daicel p35}.
+ Chiralpak WH [1424], matching SERINE {Daicel p35}.
+ Crownpak CR [1441], matching L-GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OJ [1518], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1518], matching CYCLOPENTOLATE {Daicel p21}.
+ Chiralcel OD [1582], matching PRACTOLOL {Daicel p20}.
+ Chiralcel OJ [1613], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1613], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [1855], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Crownpak CR [2009], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [2009], matching D-(+)-TRYPTOPHAN {Daicel p38}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule N-CBZ-ALANINE....
+ Chiralpak WH [219], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [219], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [361], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [361], matching N-CBZ NOR LEUCINE {Daicel p40}.
+ Chiralpak WH [507], matching N-CBZ PHENYLALANINE {Daicel p40}.
+ Chiralcel OJ [650], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [657], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [904], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [904], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.

+ Chiralcel OJ [904], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [904], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [904], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [904], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [904], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1014], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1014], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1014], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1014], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OB [1048], matching 1,3-BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1054], matching 1,4-PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1088], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [1108], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1134], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [1272], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [1272], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralcel OB [1298], matching 1,5-HEXYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1329], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1329], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1329], matching LYSINE {Daicel p38}.
+ Chiralcel OB [1348], matching 1-PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [1424], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1428], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1428], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1428], matching ALANINE {Daicel p33}.
+ Chiralcel OB [1464], matching 1-METHYL-3-PHENYL-PROPYL BENZOATE {Daicel p45}.
+ Crownpak CR [1473], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1473], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OB [1483], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1504], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1504], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1504], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OD [1513], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [1540], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1577], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1577], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1577], matching BENZOIN {Daicel p44}.
+ Chiralcel OB [1605], matching ALPHA-METHYL-PHENETHYL BENZOATE {Daicel p45}.
+ Chiralpak WM [1610], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1610], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1610], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1610], matching L-METHIONINE {Daicel p36}.
+ Chiralcel OD [1646], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Crownpak CR [1646], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1646], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1646], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1646], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1647], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralcel OK [1648], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OD [1666], matching PROPRANOLOL {Daicel p20}.
+ Crownpak CR [1668], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Chiralcel OB [1711], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak WE [1725], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1725], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1729], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1729], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1729], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1729], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak OT [1749], matching 2-ETHYL PHENYL BENZOATE {Daicel p46}.
+ Crownpak CR [1811], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OJ [1865], matching METHYL 2-PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1865], matching METHYL 2-PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OD [1931], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1931], matching PERISOXAL {Daicel p20}.
+ Chiralcel OB [2116], matching PENTYNYL-4-BENZOATE {Daicel p45}.
+ Chiralcel OB [2271], matching BUTYNYL-4-BENZOATE {Daicel p45}.
+ Chiralcel OB [2523], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralcel OB [2666], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Chiralpak WH ....
- Ordered column suggestions [column] for molecule N-CBZ-NOR-LEUCINE....
+ Chiralpak WH [70], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [70], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [191], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralpak WH [361], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [361], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OJ [1011], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [1018], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1110], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1128], matching METOPROLOL {Daicel p20}.
+ Chiralcel OB [1265], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1265], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1265], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1265], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1265], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1265], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1265], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1280], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1315], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1315], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1323], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1323], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1323], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [1375], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OG [1375], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1375], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA-1 [1375], matching 9-(2-TRIFLORO-1-HYDROXY)-ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OD [1393], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [1409], matching 1,3-BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1415], matching 1,4-PENTYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1428], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1428], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1428], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1428], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1431], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1431], matching DL-LEUCINE {Daicel p33}.
+ Chiralcel OB [1449], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [1469], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1495], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [1559], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Crownpak CR [1570], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1570], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1570], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1570], matching DL-VALINE {Daicel p33}.
+ Crownpak CR [1616], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralpak WM [1622], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1622], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1622], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1622], matching L-METHIONINE {Daicel p36}.
+ Chiralcel OB [1633], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Crownpak CR [1653], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1653], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1653], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OB [1659], matching 1,5-HEXYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1663], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.

+ Crownpak CR [1690], matching LYSINE {Daicel p38}.
+ Chiralpak WH [1690], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1690], matching LYSINE {Daicel p38}.
+ Chiralcel OK [1703], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [1709], matching 1 PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [1789], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1789], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1789], matching ALANINE {Daicel p33}.
+ Chiralcel OB [1825], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Crownpak CR [1844], matching L ORNITHINE {Daicel p38}.
+ Chiralpak WM [1834], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OB [1844], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [1966], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OB [2072], matching METHYL 3 BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [2110], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [2226], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [2226], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [2477], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [2632], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [2884], matching METHYL 2-[P (P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OB [3027], matching METHYL 2 [P (5' CHLORO 2 PYRIDYLOXY)PHENOXY] PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule N-CBZ-VALINE....
+ Chiralpak WH [70], matching N CBZ NOR LEUCINE {Daicel p40}.
+ Chiralcel OD [70], matching N CBZ NOR LEUCINE {Daicel p40}.
+ Chiralpak WH [219], matching N CBZ ALANINE {Daicel p40}.
+ Chiralcel OD [219], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralpak WH [368], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralpak WH [865], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OD [865], matching N-CBZ-ASPARAGINE {Daicel p40}.
+ Chiralcel OJ [869], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [876], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OB [1123], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1123], matching 1-(1 HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1124], matching 1 (1 HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1124], matching 1 (1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1123], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1123], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1123], matching 2-(1 HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1233], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1233], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1233], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1233], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OB [1267], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1271], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [1295], matching METOPROLOL {Daicel p20}.
+ Chiralcel OB [1307], matching 1.3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OD [1321], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OD [1326], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1353], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralcel OD [1358], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1358], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1358], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [1427], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Crownpak CR [1428], matching DL VALINE {Daicel p33}.
+ Chiralpak WH [1428], matching DL VALINE {Daicel p33}.
+ Chiralpak WM [1428], matching DL VALINE {Daicel p33}.
+ Chiralpak WE [1428], matching DL VALINE {Daicel p33}.
+ Chiralcel OD [1447], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1457], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1457], matching PERISOXAL {Daicel p20}.
+ Chiralcel OB [1491], matching 1 PHENYL PROPAN 1-OL {Daicel p42}.
+ Chiralcel OB [1517], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1547], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1547], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1547], matching LYSINE {Daicel p38}.
+ Chiralcel OB [1567], matching 1-PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [1570], matching L ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1570], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1570], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1570], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1573], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1573], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WH [1591], matching N ACETYL TRYPTOPHAN {Daicel p39}.
+ Crownpak CR [1646], matching ALANINE {Daicel p33}.
+ Chiralpak WH [1646], matching ALANINE {Daicel p33}.
+ Chiralpak WM [1646], matching ALANINE {Daicel p33}.
+ Crownpak CR [1651], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralcel OB [1683], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralpak WM [1689], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1689], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1689], matching L METHIONINE {Daicel p36}.
+ Chiralpak WE [1689], matching L METHIONINE {Daicel p36}.
+ Crownpak CR [1692], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1692], matching L-ORNITHINE {Daicel p38}.
+ Chiralcel OK [1692], matching METHYL MANDELATE {Daicel p44}.
+ Crownpak CR [1697], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [1701], matching 4-AMINO-3 (P CHLOROPHENYL)BUTYRIC ACID {Daicel p45}.
+ Chiralcel OB [1702], matching N (PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Crownpak CR [1722], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1722], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1722], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralcel OB [1823], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OB [1930], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [1968], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [2084], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [2084], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [2335], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [2490], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [2742], matching METHYL 2 [P (P' FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OB [2885], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule N-CBZ-ASPARAGINE....
+ Chiralpak WH [727], matching N-ACETYL TRYPTOPHAN {Daicel p39}.
+ Chiralpak WH [865], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [865], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralpak WH [963], matching N-ACETYL-L-PHENYLALANINE {Daicel p39}.
+ Chiralcel OD [1205], matching GUAIFENESINE {Daicel p44}.
+ Chiralpak WH [1331], matching N-ACTEYL VALINE {Daicel p39}.
+ Crownpak CR [1428], matching ASPARAGINE {Daicel p37}.
+ Chiralcel CA-1 [1512], matching MANDELAMIDE {Daicel p44}.
+ Crownpak CR [1609], matching L-GLUTAMINE {Daicel p36}.
+ Chiralcel OB [1634], matching 1-PHENYL-1,3-PROPANEDIOL {Daicel p44}.
+ Chiralpak WH [1706], matching MANDELIC ACID {Daicel p27}.
+ Chiralcel OD [1706], matching MANDELIC ACID {Daicel p27}.
+ Crownpak CR [1736], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [1736], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [1736], matching ASPARTIC ACID {Daicel p37}.
+ Chiralcel OD [1745], matching 1 PHENYL 1,2 ETHANEDIOL {Daicel p44}
+ Crownpak CR [1773], matching THREONINE {Daicel p35}
+ Chiralpak WH [1773], matching THREONINE {Daicel p35}.
+ Crownpak CR [1787], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [1787], matching HISTIDINE {Daicel p38}.
+ Crownpak CR [1825], matching SERINE {Daicel p35}.
+ Chiralpak WH [1825], matching SERINE {Daicel p35}.

+ Crownpak CR [1842], matching L GLUTAMIC ACID {Daicel p46}.
+ Chiralcel OD [1925], matching PROPOXUR {Daicel p20}.
+ Chiralcel OJ [1938], matching TRIADIMEFON {Daicel p22}.
+ Chiralpak WH [2149], matching LACTIC ACID {Daicel p27}.
+ Crownpak CR [2410], matching D-(+)-TRYPTOPHAN {Daicel p38}.
+ Chiralpak WH [2410], matching D-(+)-TRYPTOPHAN {Daicel p38}.

****

*** Testing for suggestions for column Chiralpak WH ....
- Ordered column suggestions [column] for molecule N-CBZ-PHENYLALANINE....
+ Chiralpak WH [191], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [191], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralpak WH [368], matching N CBZ VALINE {Daicel p40}.
+ Chiralcel OD [368], matching N CBZ VALINE {Daicel p40}.
+ Chiralpak WH [587], matching N CBZ ALANINE {Daicel p40}.
+ Chiralcel OD [587], matching N CBZ ALANINE {Daicel p40}.
+ Chiralcel OD [1090], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1090], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1115], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [1141], matching ALPRENOLOL {Daicel p20}.
+ Chiralpak WH [1188], matching WARFARINE {Daicel p39}.
+ Chiralcel OJ [1236], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [1243], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1267], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1299], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1299], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1299], matching BENZOIN {Daicel p44}.
+ Chiralcel OD [1348], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Crownpak CR [1428], matching D PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [1428], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [1428], matching D PHENYLALANINE {Daicel p34}.
+ Crownpak CR [1480], matching 4 AMINO 3 (P CHLOROPHENYL)BUTYRIC ACID {Daicel p45}.
+ Chiralcel OB [1490], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1490], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1490], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1490], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1490], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1490], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1490], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1509], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1515], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Crownpak CR [1592], matching PHENYLGLYCINE {Daicel p34}.
+ Chiralcel OD [1601], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel OG [1601], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralpak OT [1601], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Chiralcel CA-1 [1601], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE {Daicel p43}.
+ Crownpak CR [1638], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralpak WE [1650], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [1650], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [1653], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [1653], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1653], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [1653], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [1669], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [1669], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [1669], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [1669], matching L-METHIONINE {Daicel p36}.
+ Chiralcel OB [1674], matching 1,3-DIACETOXY-BUTANE {Daicel p45}.
+ Chiralcel OK [1679], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OD [1694], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Chiralcel OB [1721], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Crownpak CR [1795], matching DL-VALINE {Daicel p33}.
+ Chiralpak WH [1795], matching DL-VALINE {Daicel p33}.
+ Chiralpak WM [1795], matching DL-VALINE {Daicel p33}.
+ Chiralpak WE [1795], matching DL VALINE {Daicel p33}.
+ Crownpak CR [1815], matching LYSINE {Daicel p38}.
+ Chiralpak WM [1815], matching LYSINE {Daicel p38}.
+ Chiralpak WE [1815], matching LYSINE {Daicel p38}.
+ Chiralcel OB [1859], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OB [1885], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OB [1934], matching 1-PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [1959], matching L-ORNITHINE {Daicel p38}.
+ Chiralpak WM [1959], matching L-ORNITHINE {Daicel p38}.
+ Crownpak CR [2014], matching ALANINE {Daicel p33}.
+ Chiralpak WH [2014], matching ALANINE {Daicel p33}.
+ Chiralpak WM [2014], matching ALANINE {Daicel p33}.
+ Chiralcel OB [2051], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [2070], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [2191], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralcel OB [2298], matching METHYL-3-BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralpak OT [2335], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [2451], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [2451], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OB [2702], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [2857], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [3110], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE {Daicel p29}.
+ Chiralcel OB [3252], matching METHYL 2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.

****

*** Testing for suggestions for column Chiralpak WH ....
- Ordered column suggestions [column] for molecule 2-AMINO-3-PHENYLPROPIONAMIDE....
+ Crownpak CR [325], matching D PHENYLALANINE {Daicel p34}.
+ Chiralpak WH [325], matching D PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [325], matching D PHENYLALANINE {Daicel p34}.
+ Crownpak CR [434], matching 2 AMINO-2 PHENYLETHANOL {Daicel p32}.
+ Crownpak CR [447], matching 2 AMINO-1-HEXANOL {Daicel p32}.
+ Chiralpak WM [456], matching L-METHIONINE {Daicel p36}.
+ Crownpak CR [456], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WH [456], matching L-METHIONINE {Daicel p36}.
+ Chiralpak WE [456], matching L METHIONINE {Daicel p36}.
+ Chiralpak WH [489], matching N ACETYL L PHENYLALANINE {Daicel p39}.
+ Chiralpak WE [535], matching DL-LEUCINE {Daicel p33}.
+ Chiralpak WM [535], matching DL-LEUCINE {Daicel p33}.
+ Crownpak CR [535], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WH [535], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WM [535], matching L-ISOLEUCINE {Daicel p34}.
+ Chiralpak WE [535], matching L-ISOLEUCINE {Daicel p34}.
+ Crownpak CR [647], matching 2-AMINO-1-PENTANOL {Daicel p32}.
+ Crownpak CR [654], matching DL-TYROSINE {Daicel p35}.
+ Crownpak CR [744], matching LYSINE {Daicel p38}.
+ Chiralpak WM [744], matching LYSINE {Daicel p38}.
+ Chiralpak WE [744], matching LYSINE {Daicel p38}.
+ Crownpak CR [786], matching 2-AMINO-1-PHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [821], matching N-(PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [900], matching 2-ACETOXY-3-PHENYLPROPANE {Daicel p45}.
+ Chiralpak WH [931], matching N-AMIDE-L-PHENYLALANINE {Daicel p41}.
+ Crownpak CR [945], matching ERYTHRO-2-AMINO-1,2-DIPHENYLETHANOL {Daicel p32}.
+ Chiralcel OB [974], matching 2-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OK [1007], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [1132], matching 1-PHENYLETHANOL {Daicel p42}.
+ Crownpak CR [1148], matching L-3,4-DIHYDROXYPHENYLALANINE {Daicel p35}.
+ Chiralcel OB [1208], matching 1-PHENYL-PROPAN-1-OL {Daicel p42}.
+ Chiralcel OD [1230], matching CARBINOXAMINE {Daicel p21}.
+ Chiralcel OB [1243], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1243], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1243], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1243], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1243], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1243], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1243], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.

+ Chiralcel OD [1279], matching BENZYL PHENYL CARBINOL [Daicel p46].
+ Chiralcel OD [1309], matching BENZOIN [Daicel p44].
+ Chiralcel OF [1309], matching BENZOIN [Daicel p44].
+ Chiralpak OP [1309], matching BENZOIN [Daicel p44].
+ Chiralcel OD [1332], matching 2PHENOXYPROPIONIC ACID [Daicel p23].
+ Chiralcel OF [1332], matching 2PHENOXYPROPIONIC ACID [Daicel p26].
+ Crownpak CR [1376], matching ARGININE [Daicel p39].
+ Chiralpak WE [1376], matching ARGININE [Daicel p39].
+ Chiralpak WM [1376], matching ARGININE [Daicel p39].
+ Crownpak CR [1479], matching CITRULLINE [Daicel p37].
+ Chiralcel OJ [1540], matching METHYL 2 PHENOXYPROPIONATE [Daicel p29].
+ Chiralcel OK [1540], matching METHYL 2 PHENOXYPROPIONATE [Daicel p29].
+ Chiralcel OD [1559], matching PERISOXAL [Daicel p20].
+ Chiralcel OF [1559], matching PERISOXAL [Daicel p20].
+ Chiralcel OD [1572], matching ALPRENOLOL [Daicel p20].
+ Chiralcel OD [1580], matching PROPRANOLOL [Daicel p20].
+ Chiralcel OB [1625], matching N,O-DIBENZOYL2-AMINOBUTANOL [Daicel p46].
+ Chiralcel OD [1631], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE [Daicel p43].
+ Chiralcel OG [1631], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE [Daicel p43].
+ Chiralpak OT [1631], matching 9 (2 TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE [Daicel p43].
+ Chiralcel CA-1 [1631], matching 9 (2 TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE [Daicel p43].
+ Chiralcel OB [1658], matching BUTYNYL 3 BENZOATE [Daicel p45].
+ Chiralcel OB [1783], matching PENTYNYL 3 BENZOATE [Daicel p45].
+ Chiralcel OD [1836], matching METOPROLOL [Daicel p20].
+ Chiralcel OD [1898], matching 2 BUTYL PHENYL CARBAMATE [Daicel p46].
+ Chiralcel OJ [2005], matching CYCLOPENTOLATE [Daicel p21].
+ Chiralcel OD [2005], matching CYCLOPENTOLATE [Daicel p21].
+ Chiralcel OD [2634], matching CARTEOLOL [Daicel p20].

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule THREONINE....
+ Crownpak CR [73], matching SERINE [Daicel p35].
+ Chiralpak WH [73], matching SERINE [Daicel p35].
+ Crownpak CR [92], matching ASPARTIC ACID [Daicel p37].
+ Chiralpak WH [92], matching ASPARTIC ACID [Daicel p37].
+ Chiralpak WM [92], matching ASPARTIC ACID [Daicel p37].
+ Crownpak CR [254], matching L-GLUTAMIC ACID [Daicel p36].
+ Crownpak CR [268], matching ASPARAGINE [Daicel p37].
+ Crownpak CR [364], matching L-GLUTAMINE [Daicel p36].
+ Crownpak CR [434], matching HISTIDINE [Daicel p38].
+ Chiralpak WH [434], matching HISTIDINE [Daicel p38].
+ Chiralpak WH [439], matching GLYCERIC ACID [Daicel p27].
+ Crownpak CR [471], matching ALANINE [Daicel p33].
+ Chiralpak WH [471], matching ALANINE [Daicel p33].
+ Chiralpak WM [471], matching ALANINE [Daicel p33].
+ Crownpak CR [514], matching DL-VALINE [Daicel p33].
+ Chiralpak WH [514], matching DL-VALINE [Daicel p33].
+ Chiralpak WM [514], matching DL-VALINE [Daicel p33].
+ Chiralpak WE [514], matching DL-VALINE [Daicel p33].
+ Crownpak CR [666], matching 2-AMINO-1-PENTANOL [Daicel p32].
+ Crownpak CR [707], matching D (+) TRYPTOPHAN [Daicel p38].
+ Chiralpak WH [707], matching D (+) TRYPTOPHAN [Daicel p38].
+ Chiralpak WH [808], matching LACTIC ACID [Daicel p27].
+ Crownpak CR [962], matching L 3,4 DIHYDROXYPHENYLALANINE [Daicel p35].
+ Crownpak CR [1019], matching 2 AMINO BUTANE [Daicel p32].
+ Chiralpak WH [1027], matching N-ACTEYL VALINE [Daicel p39].
+ Chiralcel OB [1068], matching 1-PHENYL-1,3-PROPANEDIOL [Daicel p44].
+ Chiralcel OB [1140], matching 1 PHENYL-1,2 ETHANEDIOL [Daicel p44].
+ Chiralpak WH [1200], matching MANDELIC ACID [Daicel p27].
+ Chiralcel OD [1200], matching MANDELIC ACID [Daicel p27].
+ Chiralpak WH [1294], matching N ACETYL L PHENYLALANINE [Daicel p39].
+ Chiralcel OD [1406], matching GUAIFENESINE [Daicel p44].
+ Chiralcel CA-1 [1437], matching MANDELAMIDE [Daicel p44].
+ Chiralpak WH [1695], matching N-CBZ-ASPARAGINE [Daicel p40].
+ Chiralcel OD [1695], matching N CBZ ASPARAGINE [Daicel p40].
+ Chiralcel OD [1809], matching PINDOLOL [Daicel p20].
+ Chiralcel OJ [1886], matching TRIADIMEFON [Daicel p22].
+ Chiralcel OD [2120], matching METOPROLOL [Daicel p20].
+ Chiralcel OD [2318], matching PRACTOLOL [Daicel p20].

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule SERINE....
+ Crownpak CR [73], matching THREONINE [Daicel p35].
+ Chiralpak WH [73], matching THREONINE [Daicel p35].
+ Crownpak CR [162], matching ASPARTIC ACID [Daicel p37].
+ Chiralpak WH [162], matching ASPARTIC ACID [Daicel p37].
+ Chiralpak WM [162], matching ASPARTIC ACID [Daicel p37].
+ Chiralpak WH [344], matching GLYCERIC ACID [Daicel p27].
+ Crownpak CR [366], matching ASPARAGINE [Daicel p37].
+ Crownpak CR [403], matching L-GLUTAMIC ACID [Daicel p36].
+ Crownpak CR [500], matching ALANINE [Daicel p33].
+ Chiralpak WH [500], matching ALANINE [Daicel p33].
+ Chiralpak WM [500], matching ALANINE [Daicel p33].
+ Crownpak CR [577], matching L-GLUTAMINE [Daicel p36].
+ Crownpak CR [600], matching HISTIDINE [Daicel p38].
+ Chiralpak WH [600], matching HISTIDINE [Daicel p38].
+ Chiralpak WH [723], matching LACTIC ACID [Daicel p27].
+ Crownpak CR [861], matching D-(+)-TRYPTOPHAN [Daicel p38].
+ Chiralpak WH [861], matching D-(+)-TRYPTOPHAN [Daicel p38].
+ Chiralpak WH [1045], matching MANDELIC ACID [Daicel p27].
+ Chiralcel OD [1045], matching MANDELIC ACID [Daicel p27].
+ Chiralcel OB [1068], matching 1-PHENYL-1,2-ETHANEDIOL [Daicel p44].
+ Crownpak CR [1111], matching L-3,4-DIHYDROXYPHENYLALANINE [Daicel p35].
+ Chiralpak WH [1125], matching N ACTEYL VALINE [Daicel p39].
+ Crownpak CR [1141], matching 2 AMINO BUTANE [Daicel p32].
+ Chiralcel OB [1223], matching 1-PHENYL-1,3 PROPANEDIOL [Daicel p44].
+ Chiralcel OD [1334], matching GUAIFENESINE [Daicel p44].
+ Chiralcel CA-1 [1347], matching MANDELAMIDE [Daicel p44].
+ Chiralpak WH [1393], matching N-ACETYL-L-PHENYLALANINE [Daicel p39].
+ Chiralpak WH [1793], matching N-CBZ-ASPARAGINE [Daicel p40].
+ Chiralcel OD [1793], matching N-CBZ-ASPARAGINE [Daicel p40].
+ Chiralcel OD [1942], matching PINDOLOL [Daicel p20].
+ Chiralcel OJ [2086], matching TRIADIMEFON [Daicel p22].
+ Chiralcel OD [2189], matching PRACTOLOL [Daicel p20].
+ Chiralcel OD [2202], matching METOPROLOL [Daicel p20].

****

*** Testing for suggestions for column Chiralpak WH ....
= Ordered column suggestions [column] for molecule ALANINE....
+ Crownpak CR [219], matching DL-VALINE [Daicel p33].
+ Chiralpak WH [219], matching DL-VALINE [Daicel p33].
+ Chiralpak WM [219], matching DL-VALINE [Daicel p33].
+ Chiralpak WE [219], matching DL-VALINE [Daicel p33].
+ Chiralpak WH [344], matching LACTIC ACID [Daicel p27].
+ Crownpak CR [352], matching 2-AMINO-1-PENTANOL [Daicel p32].
+ Crownpak CR [361], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WH [361], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WM [361], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WE [361], matching L-ISOLEUCINE [Daicel p34].
+ Chiralpak WE [365], matching DL-LEUCINE [Daicel p34].
+ Chiralpak WM [365], matching DL-LEUCINE [Daicel p34].
+ Crownpak CR [443], matching PHENYLGLYCINE [Daicel p34].
+ Crownpak CR [471], matching THREONINE [Daicel p35].
+ Chiralpak WH [471], matching THREONINE [Daicel p35].
+ Chiralpak WM [480], matching L-METHIONINE [Daicel p36].
+ Crownpak CR [480], matching L-METHIONINE [Daicel p36].
+ Chiralpak WH [480], matching L-METHIONINE [Daicel p36].
+ Chiralpak WE [480], matching L-METHIONINE [Daicel p36].
+ Crownpak CR [494], matching 2-AMINO-1-HEXANOL [Daicel p32].
+ Crownpak CR [500], matching SERINE [Daicel p35].
+ Chiralpak WH [500], matching SERINE [Daicel p35].
+ Crownpak CR [587], matching D PHENYLALANINE [Daicel p34].

+ Chiralpak WH [587], matching D-PHENYLALANINE {Daicel p34}.
+ Chiralpak WM [587], matching D PHENYLALANINE {Daicel p34}.
+ Crownpak CR [592], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WH [592], matching ASPARTIC ACID {Daicel p37}.
+ Chiralpak WM [592], matching ASPARTIC ACID {Daicel p37}.
+ Crownpak CR [621], matching L ORNITHINE {Daicel p38}.
+ Chiralpak WM [621], matching L ORNITHINE {Daicel p38}.
+ Crownpak CR [642], matching 2 AMINO BUTANE {Daicel p32}.
+ Crownpak CR [666], matching L GLUTAMIC ACID {Daicel p36}.
+ Chiralcel OB [674], matching 1,3 DIACETOXY BUTANE {Daicel p45}.
+ Crownpak CR [766], matching LYSINE {Daicel p38}.
+ Chiralpak WM [766], matching LYSINE {Daicel p38}.
+ Chiralpak WE [766], matching LYSINE {Daicel p38}.
+ Crownpak CR [775], matching 4-AMINO-3-(P-CHLOROPHENYL)BUTYRIC ACID
{Daicel p35}.
+ Crownpak CR [831], matching 2-AMINO-1-PHENYLETHANOL {Daicel p42}.
+ Chiralpak WH [844], matching GLYCERIC ACID {Daicel p27}.
+ Crownpak CR [857], matching L-GLUTAMINE {Daicel p36}.
+ Chiralcel OD [863], matching 2-PHENYLPROPIONIC ACID {Daicel p27}.
+ Crownpak CR [891], matching HISTIDINE {Daicel p38}.
+ Chiralpak WH [891], matching HISTIDINE {Daicel p38}.
+ Chiralcel OJ [1006], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1006], matching METHYL 2 PHENOXYPROPIONATE {Daicel p29}.
+ Chiralcel OK [1063], matching METHYL MANDELATE {Daicel p44}.
+ Chiralcel OB [1068], matching 1 PHENYLETHANOL {Daicel p42}.
+ Chiralcel OB [1078], matching METHYL 3 BENZOYLOXYPROPANOATE {Daicel p45}.
+ Chiralcel OB [1095], matching BUTYNYL 3 BENZOATE {Daicel p45}.
+ Chiralcel OB [1103], matching 2 ACETOXY 3 PHENYLPROPANE {Daicel p45}.
+ Chiralcel OB [1144], matching 1 PHENYL PROPAN 1 OL {Daicel p42}.
+ Chiralcel OB [1219], matching N (PHENYLETHYL)ACETAMIDE {Daicel p46}.
+ Chiralcel OB [1249], matching PENTYNYL 3 BENZOATE {Daicel p45}.
+ Crownpak CR [1271], matching CITRULLINE {Daicel p37}.
+ Chiralpak WH [1428], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OD [1428], matching N-CBZ-ALANINE {Daicel p40}.
+ Chiralcel OJ [1438], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1438], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OK [1438], matching 2-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OB [1438], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OC [1438], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OJ [1438], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralcel OD [1438], matching 1-(1-HYDROXYETHYL)NAPHTHALENE {Daicel p43}.
+ Chiralpak OT [1497], matching 2 ETHYL PHENYL BENZOATE {Daicel p46}.
+ Chiralcel OJ [1497], matching FENOPROFEN {Daicel p27}.
+ Chiralcel OJ [1537], matching KETOPROFEN {Daicel p27}.
+ Chiralcel OD [1607], matching BENZOIN {Daicel p44}.
+ Chiralcel OF [1607], matching BENZOIN {Daicel p44}.
+ Chiralpak OP [1607], matching BENZOIN {Daicel p44}.
+ Chiralcel OB [1618], matching METHYL 2-[P-(P'-FLOROPHENOXY)PHENOXY]PROPIONATE
{Daicel p29}.
+ Chiralpak WH [1646], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1646], matching N-CBZ-VALINE {Daicel p40}.
+ Chiralcel OD [1654], matching BENZYL PHENYL CARBINOL {Daicel p46}.
+ Chiralcel OB [1661], matching ALPHA METHYL PHENETHYL BENZOATE {Daicel p45}.
+ Chiralpak WH [1789], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OD [1789], matching N-CBZ-NOR-LEUCINE {Daicel p40}.
+ Chiralcel OB [1801], matching 1 METHYL 3 PHENYL PROPYL BENZOATE {Daicel p45}.
+ Chiralcel OB [1807], matching METHYL
2-[P-(5'-CHLORO-2-PYRIDYLOXY)PHENOXY]-PROPIONATE {Daicel p29}.
+ Chiralcel OB [1831], matching 1,3 BUTANEDIOL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [1836], matching 9 (2-TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OC [1836], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralpak OT [1836], matching 9 (2-TRIFLORO 1-HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel CA 1 [1836], matching 9 (2 TRIFLORO 1 HYDROXY) ETHYL ANTHRACENE
{Daicel p43}.
+ Chiralcel OD [1876], matching PERISOXAL {Daicel p20}.
+ Chiralcel OF [1876], matching PERISOXAL {Daicel p20}.
+ Chiralcel OD [1957], matching PROPRANOLOL {Daicel p20}.
+ Chiralcel OD [1975], matching ALPRENOLOL {Daicel p20}.
+ Chiralcel OB [1977], matching 1,4 PENTYL DIBENZOATE {Daicel p45}.
+ Chiralpak WH [2014], matching N-CBZ-PHENYLALANINE {Daicel p40}.
+ Chiralcel OB [2127], matching 1,5 HEXYL DIBENZOATE {Daicel p45}.
+ Chiralcel OD [2161], matching METOPROLOL {Daicel p20}.
+ Chiralcel OD [2609], matching CARTEOLOL {Daicel p20}.

YES

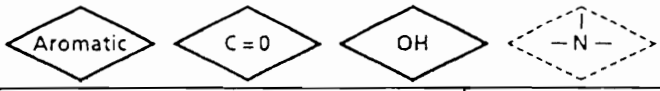
STAUFFERST job terminated at 14-AUG-1993 16:22:49.61

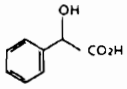
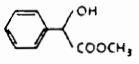
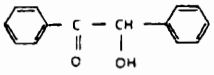
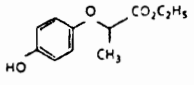
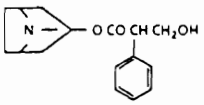
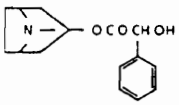
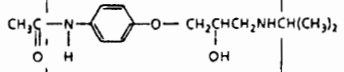
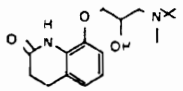
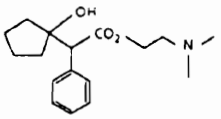
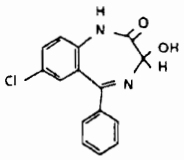
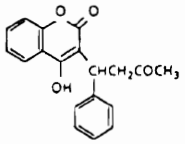
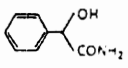
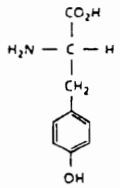
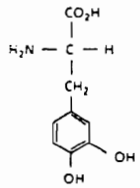
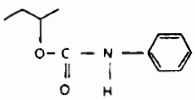
Accounting information:

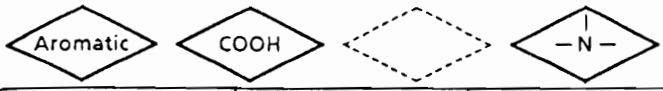
Buffered I/O count:	689	Peak working set size:	4096
Direct I/O count:	2224	Peak page file size:	19787
Page faults:	19928	Mounted volumes:	0
Charged CPU time:	0 04:31:44.51	Elapsed time:	0 04:35:16.00

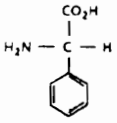
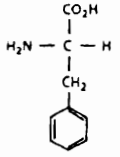
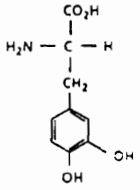
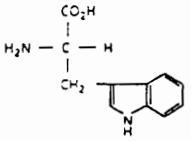
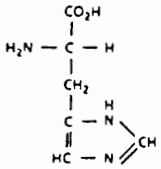
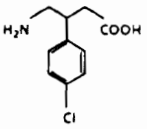
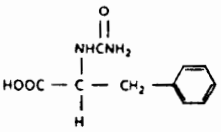
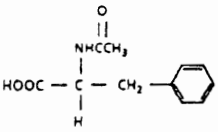
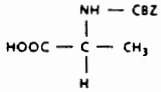
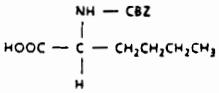
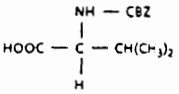
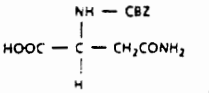
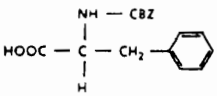
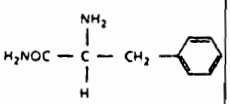
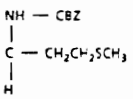
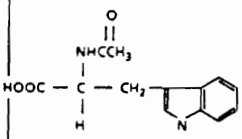
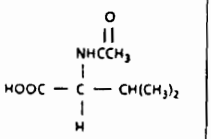
APPENDIX J: Daicel Chiral Column Selection Guide

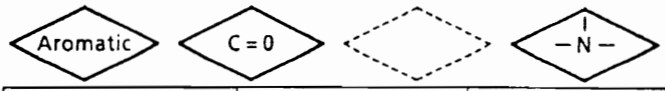
Appendix J contains an excerpt from the Daicel Chiral Column Applications Guide[111] that is pertinent to OPTIMER. The result of OPTIMER is to suggest a page number of where to go to look for similar molecules to your Target Molecule. This appendix lists those pages. Figure 7[111] lists the decision tree on which OPTIMER is based. Section IV.B describes OPTIMER.

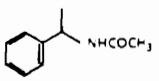
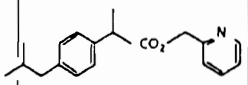
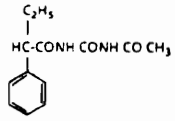
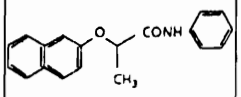
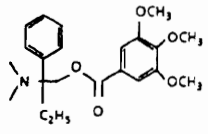
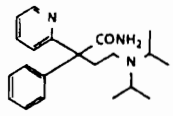
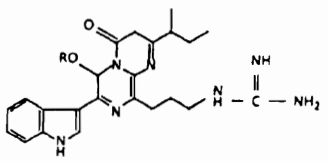
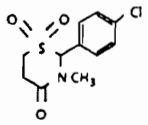
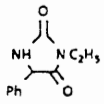
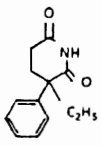
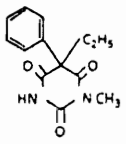
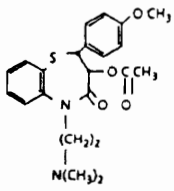
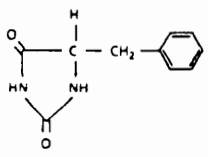
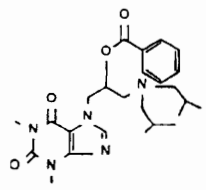
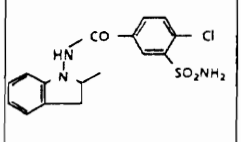
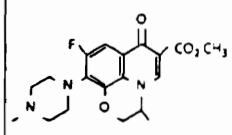
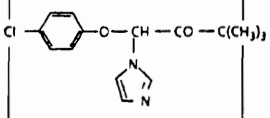
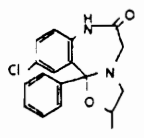


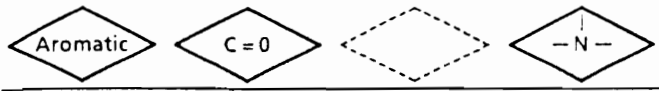
			
WH, OD →P.27	OK, OJ →P.44	OP, OD, OF →P.44	OJ →P.29
			
OD →P.21	OD →P.21	OD →P.20	OD →P.20
			
OJ, OD →P.21	OG, OF →P.25	OC →P.22	CA-1 →P.44
			
CR →P.35	CR →P.35	OD →P.46	

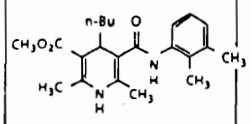
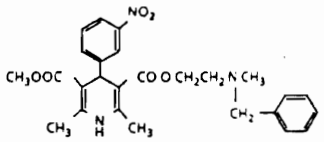
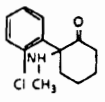
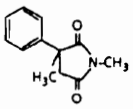
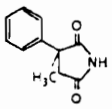
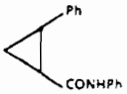
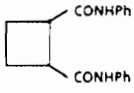
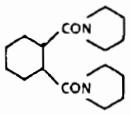
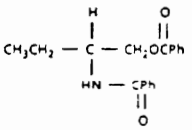
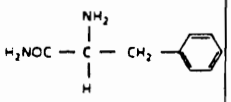


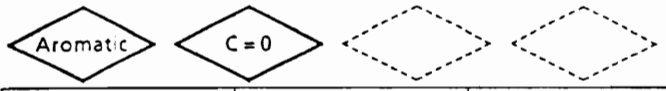
			
CR →P.34	CR, WH, WM →P.34	CR →P.35	CR, WH →P.37
			
CR, WH →P.38	CR →P.35	WH →P.41	WH →P.39
			
WH, OD →P.40	WH, OD →P.40	WH, OD →P.40	WH, OD →P.40
			F — MOC — Aminoacid
WH →P.40	WH →P.41	WH →P.40	OD
			
WH →P.39	WH →P.39		

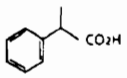
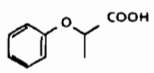
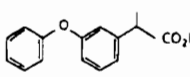
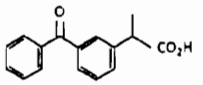
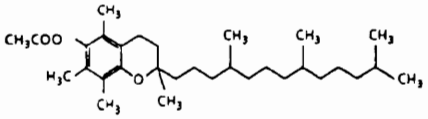
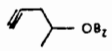
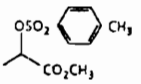
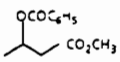
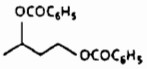
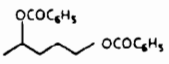
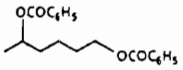
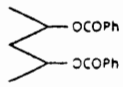
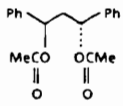
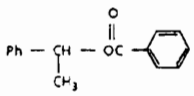
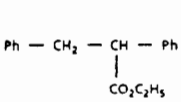
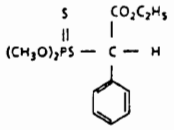
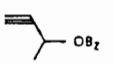
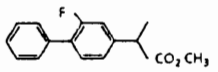
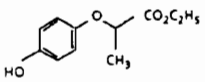


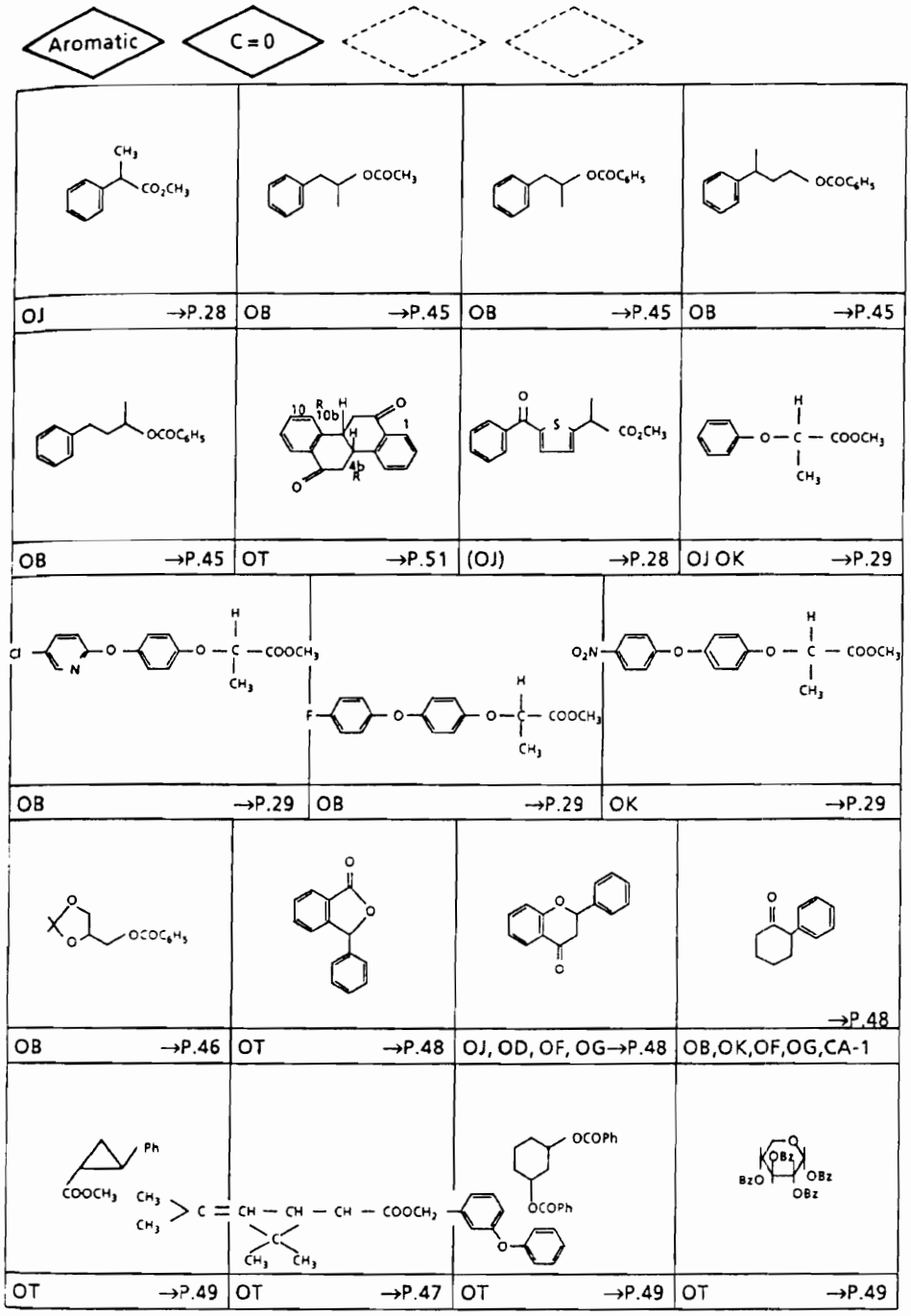
			
OB →P.46	OD →P.28	OF, OG →P.28	OJ, CA-1, OB →P.29
			
OJ →P.28	OF →P.21	OT →P.26	
			
OJ →P.23	OJ, CA-1 →P.24	OJ, CA-1 →P.24	OJ, CA-1 →P.24
			
OF, OG →P.25	WH →P.41	(OF) (OJ)	(OJ) →P.41
			
(OD)		OJ →P.22	OF →P.25

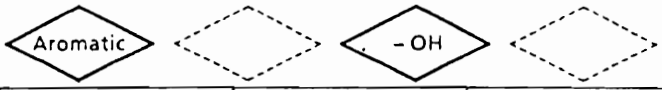


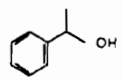
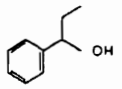
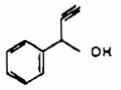
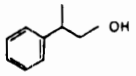
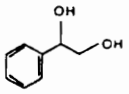
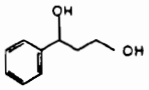
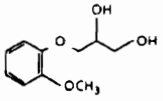
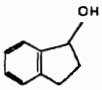
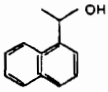
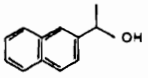
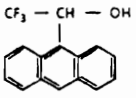
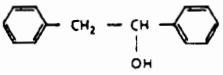
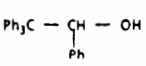
			
OF →P.26	(OF) →P.26		
			
CA-1 →P.48	OJ →P.24	OJ →P.24	
			
OT →P.49	OT →P.49	OT →P.49	OT →P.49
			
OB →P.46			
			
WH →P.41			

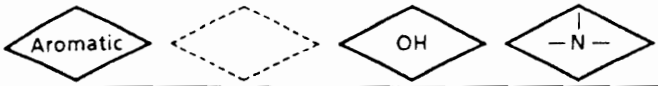


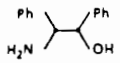
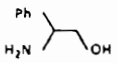
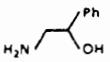
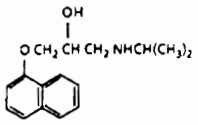
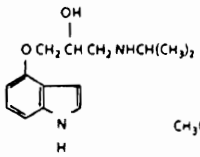
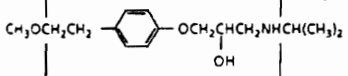
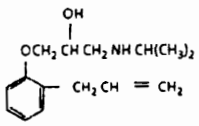
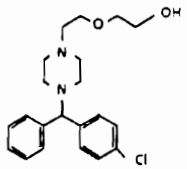
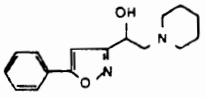
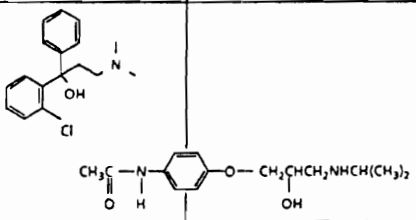
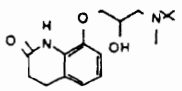
			
OD →P.27	OD→P.29, OF→P.26	OJ →P.27	OJ →P.27
			
OP →P.26		OB →P.45	OB
			
OB →P.45	OB →P.45	OB →P.45	OB →P.45
			
OT →P.46	OT →P.46	OT →P.46	OT →P.46
			
OB, OJ →P.30	OB →P.45	OJ →P.28	OJ →P.29

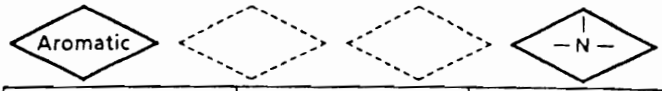

Aromatic
C = 0
OJ →P.28
OB →P.45
OB →P.45
OB →P.45
OB →P.45
OT →P.51
(OJ) →P.28
OJ OK →P.29
OB →P.29
OB →P.29
OK →P.29
OB →P.46
OT →P.48
OJ, OD, OF, OG →P.48
→P.48
OB,OK,OF,OG,CA-1
OT →P.49
OT →P.47
OT →P.49
OT →P.49

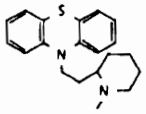
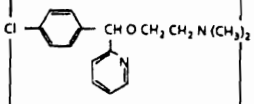
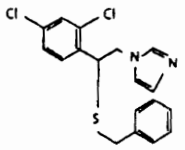
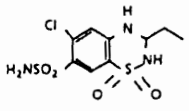
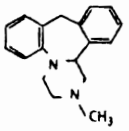
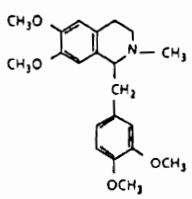
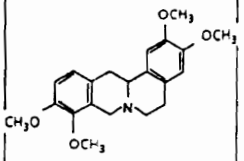


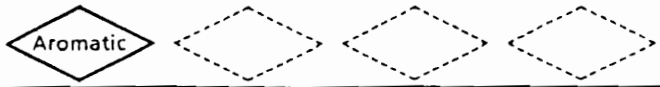
			
OB →P.42	OB →P.42	OK, OJ →P.42	OB →P.42
			
OB →P.44	OB →P.44	OD →P.44	OB, OJ →P.42
			
OB, OJ →P.42	OB →P.42	OB, OC, OJ, OD→P.43	OJ, OB, OK →P.43
 →P.43			
OD,OG,OT,CA-1	OD →P.46	OT →P.46	

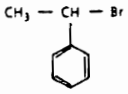
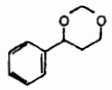


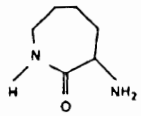
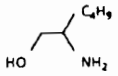
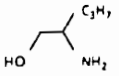
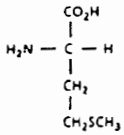
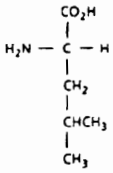
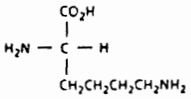
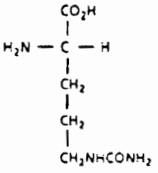
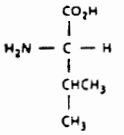
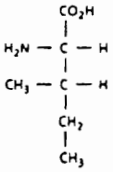
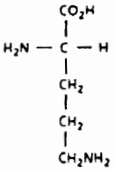
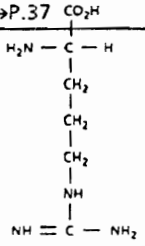
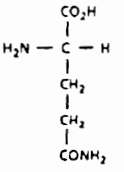
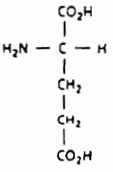
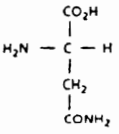
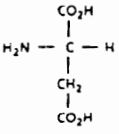
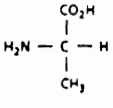
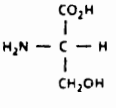
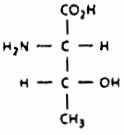
			
CR →P.32	CR →P.32	CR →P.32	
	 		
OD →P.20	OD →P.20	OD →P.20	OD →P.20
			
OD, OJ →P.22	OD, OF →P.20	OJ (OD) →P.21	OD →P.20
			
OD →P.20			

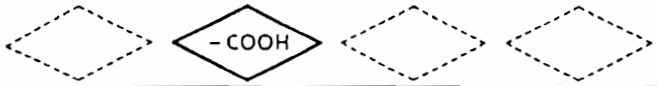


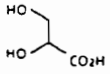
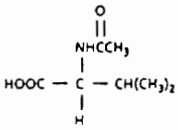
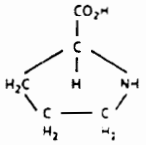
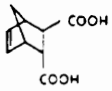
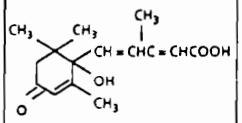
			
CR →P.32	(OJ) →P.22	OD →P.22	OD (OJ) →P.22
			
OF →P.23	OJ →P.26	OD →P.26	OC, OD →P.26

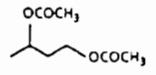
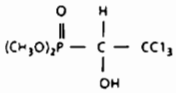
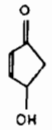
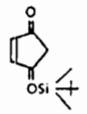
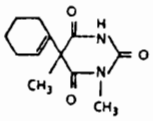
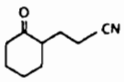
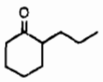


			
OJ, OT, CA-1 →P.49	OT →P.46	OB→P.48,OA,OB→P.49	OT,OB,OC,OD →P.47

			
CR →P.32	CR →P.32	CR →P.32	CR →P.32
 →P.36			
CR, WM, (WH) (WE)	WE, WM →P.33	CR, WM, WE →P.38	CR →P.37
 →P.33	 →P.34		
CR, WH, WM, WE	CR, WH, WM, WE	CR, WM →P.38	CR, WE →P.39
			
CR →P.36	CR →P.36	CR →P.36	CR, WH (WM) →P.37
			
(CR)	CR, WE →P.33	CR, WH →P.35	CR, WH →P.35



			
(WH) →P.27	WH →P.27	WH →P.39	WH →P.39
			
OD →P.27	OD →P.27	OD →P.27	

			
OB →P.45	(OB) →P.45		
			
OA →P.47	OA →P.47	OA →P.47	OC
			
CA-1 →P.24	OB →P.48	OB →P.48	OB →P.48