The Virginia Beef Cattle Simulation Model: A bio-economic simulation program modeling the interactions among reproduction, forage availability, nutrition, growth, and marketing in beef cattle.

by

James Henry Schick

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 Animal Science (Animal Breeding and Genetics)

W. D. Hohenboken, Chairman
D. R. Notter
W. D. Purcell
W. D. Whittier
B. R. McKinnon

March, 1999
Blacksburg, Virginia

Key Words: Reproduction, Growth, Forage, Marketing, Simulation Model, Beef Cattle
Copyright 1999. James Henry Schick
The Virginia Beef Cattle Simulation Model: A bio-economic simulation program modeling the interactions among reproduction, forage availability, nutrition, growth, and marketing in beef cattle.

by
James Henry Schick

W. D. Hohenboken, Chairman

Department Of Animal and Poultry Sciences

(ABSTRACT)

The Virginia Beef Cattle Simulation Model (VBCSM) is a user-friendly, dynamic, stochastic computer program whose objective is to serve as a decision-aid for Virginia cattlemen dealing with complex management issues such as whether to retain weaned calves through the stocker growth stage. Its five source-code modules are reproduction, forage, nutrition, marketing, and a tool that randomly assigns values to variables from appropriate statistical distributions. The VBCSM contains production statistics for 12 breeds, 21 forage species, and three Virginia agro-ecological zones. It simulates at the animal level using information obtained from program dialog. Help can be activated on each dialog page. It is event-driven on a daily time increment. The reproduction module simulates puberty, conception, abortion, parturition, dystocia, lactation, pregnancy testing, culling, within-herd replacement female
selection, open or pregnant replacement female purchases, cow and calf mortality, and weaning. The forage module simulates daily pasture growth dependent upon month, precipitation, erosion, pasture maintenance, grazing system, farm location, weed infestation, and slope. This module interacts with the nutrition module to calculate each animal’s forage intake, supplemental feed requirements, and daily gain or loss using National Research Council equations. The marketing routine sells the weanling calves to the stocker herd and sells stocker calves, orphan calves, and cull cows through user-specified markets, including the Virginia Tel-O-Market auction. After simulating for eight years to achieve equilibrium conditions, the VBCSM provides an income statement for the cow-calf operation and a partial budget for net income or loss from the stocker herd for up to three years. VBCSM was rigorously tested using a mathematical model with two calving seasons, three lengths of breeding season, four culling policies, and a year effect. Descriptive statistics suggest that the program code works in a consistent manner. However, several potential programming inconsistencies were discovered. Simulation results indicate that fall calving may be more profitable for Virginia cattle producers than spring calving for weanling calf production, but a spring calved stocker program may be more profitable that a fall calved stocker program. Perhaps, VBCSM will help cattlemen to enhance their
profits by more efficient market planning and utilization of production resources.
In Fond Memory,

Jared Bronson Schick

June 11, 1971 – August 3, 1988

“He is a child of the King”
Acknowledgements

I would like to thank the members of my graduate committee, Drs. W. D. Hohenboken, Dr. R. Notter, W. D. Purcell, W. D. Whittier, and B. R. McKinnon for their kind and patient guidance, counsel, and encouragement during the course of my work at Virginia Tech. Specifically, I extend my thanks to Dr. W. D. Hohenboken for serving as chairman of my dissertation committee and for sharing his knowledge and experience. Also, I specifically extend my thanks to Dr. W. D. Whittier for "freighting" me around the state and giving me his wise and personal insights into the Virginia cattle business.

I am also deeply grateful to Drs. Gregory S. Lewis, Richard G. Saacke, and W. E. Beal for their technical assistance in modeling many stages of my project. I truly appreciate their generous efforts.

I am also indebted to the many faculty members that have helped my through my course work such as Drs. J. W. Knight, Ina Hoeschele, R. E. Pearson, Eric Wong, and, of course, the unforgettable Paul B. Siegel.

I would also like to express my sincere appreciation to my programming instructors and friends, Jonathan Turner and Claire Chen for their assistance and efforts in helping me to work through the programming and debugging stages of this
modeling project.

A special thanks is also extended to my fellow graduate students for their assistance and friendship during the course of my graduate studies. In particular, I would like to acknowledge Kwame Boa-Amponsem and the jocular Larry Wilson.

Lastly, but certainly not least, I wish to thank my wife, Ann Frances, and my teenage son, Robert Hugh, without whose love, understanding, and support, I could not have pursued my goals.
# Table of Contents

<table>
<thead>
<tr>
<th>Content</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1. General Background</td>
<td>2</td>
</tr>
<tr>
<td>1.2. Objectives Of The Study</td>
<td>6</td>
</tr>
<tr>
<td>Literature Cited</td>
<td>7</td>
</tr>
<tr>
<td>CHAPTER 2. Literature Review</td>
<td>9</td>
</tr>
<tr>
<td>Abstracts</td>
<td>10</td>
</tr>
<tr>
<td>2.1. Introduction</td>
<td>10</td>
</tr>
<tr>
<td>2.2. Introduction to Operations Research</td>
<td>11</td>
</tr>
<tr>
<td>2.3. Simulation in Beef Cattle Production</td>
<td>12</td>
</tr>
<tr>
<td>2.4. Deterministic vs Stochastic Simulation Models</td>
<td>18</td>
</tr>
<tr>
<td>2.5. Dynamic vs Static Simulation Models</td>
<td>19</td>
</tr>
<tr>
<td>2.6. Risk Management in Simulation Modeling</td>
<td>19</td>
</tr>
<tr>
<td>2.7. C++ &amp; Object-Oriented Programming</td>
<td>20</td>
</tr>
<tr>
<td>2.8. A Program’s Lifecycle</td>
<td>22</td>
</tr>
<tr>
<td>2.9. Debugging or Verifying Simulation Models</td>
<td>23</td>
</tr>
<tr>
<td>2.10. Validating Simulation Models</td>
<td>25</td>
</tr>
<tr>
<td>2.11. Beef Cattle Simulation Models</td>
<td>26</td>
</tr>
<tr>
<td>2.11.1. Colorado Model</td>
<td>26</td>
</tr>
<tr>
<td>2.11.2. HERDPLAN</td>
<td>28</td>
</tr>
<tr>
<td>2.11.3. Kentucky Beef-Forage Model</td>
<td>29</td>
</tr>
<tr>
<td>2.11.4. Notter Model</td>
<td>31</td>
</tr>
<tr>
<td>2.11.5. Spur Model</td>
<td>31</td>
</tr>
</tbody>
</table>
2.11.6. Texas A&M Simulation Model

2.12. Conclusion

Literature Cited

Chapter 3. The Virginia Beef Cattle Simulation Model

Abstract

3.1. Introduction

3.2. The Virginia Beef Cattle Simulation Model

3.2.1. Forage and Nutrition

3.2.2. Reproduction and Life Events

3.2.3. Marketing

3.2.4. Tools

LITERATURE CITED

Chapter 4. Reproduction and Life Events Module

Abstract

4.1. Introduction

4.2. UseInput1

4.3. UseInput2

4.4. UseInput11

4.5. UseInput2A

4.6. UseInput2B

4.7. CreateHerd

4.8. GetHeadAnimal

4.9. HerdHead
| 4.10. | ShrinkAnimal | 76 |
| 4.11. | GetAnimal | 78 |
| 4.12. | Dystocia | 78 |
| 4.13. | Sterility | 80 |
| 4.14. | Parturition | 81 |
| 4.15. | CalfMilk | 84 |
| 4.16. | Puberty | 84 |
| 4.17. | PregnancyTest | 86 |
| 4.18. | CullCows | 88 |
| 4.19. | Lactation | 89 |
| 4.20. | BreedFemales | 90 |
| 4.22. | AddAnimal | 100 |
| 4.23. | LastID | 100 |
| 4.24. | Abort | 101 |
| 4.25. | Mortality | 103 |
| 4.26. | SBBreeders | 105 |
| 4.27. | GetSex | 109 |
| 4.28. | Wean | 109 |
| 4.29. | ChangeSex | 110 |
| 4.30. | ReproStats | 111 |
| 4.31. | DeadCattle | 111 |
| 4.32. | DeleteAnimal | 112 |
| 4.33. | DumpList | 112 |
| 4.34. | ResetCowVariables | 112 |
| 4.35. | UseYearOption | 113 |
4.36. DeleteHerd........................................................................................................ 113
4.37. ResetCowConCntr............................................................................................ 113

Literature Cited........................................................................................................ 114

Chapter 5. Forage and Nutrition Modules................................................................. 121

Abstract.................................................................................................................... 122

5.1. Introduction........................................................................................................ 122

5.2. FORAGE.............................................................................................................. 122

5.2.1. UseInput5....................................................................................................... 122

5.2.2. UseInput9....................................................................................................... 123

5.2.3. UseInput10.................................................................................................... 124

5.2.4. UseInput6....................................................................................................... 124

5.2.5. MainForage.................................................................................................... 124

5.2.6. ResetForageConCntr...................................................................................... 132

5.2.7. ResetForageVariables.................................................................................. 132

5.2.8. Temperature.................................................................................................. 133

5.3. NUTRITION....................................................................................................... 133

5.3.1. UseInput7....................................................................................................... 134

5.3.2. UseInput8....................................................................................................... 134

5.3.3. MainNutrition................................................................................................ 135

5.3.4. DWGAIN......................................................................................................... 140

5.3.6. FeedUsage...................................................................................................... 154

5.3.7. ResetNutritionVariables.............................................................................. 155

Literature Cited........................................................................................................ 157

Chapter 6. Marketing Module.................................................................................... 176

Abstract.................................................................................................................... 177
6.1. Introduction........................................................................................................ 177
6.2. Marketing Alternatives in Virginia................................................................. 177
  6.2.1. Marketing Methods...................................................................................... 177
  6.2.2. Auction Marketing...................................................................................... 178
  6.2.3. Direct Marketing......................................................................................... 179
  6.2.4. Electronic Marketing................................................................................... 180
          6.2.4.1 Virginia Tel-O-Market
          Auction........................................................................................................ 181
  6.2.5. Futures Markets........................................................................................ 182
6.3. UseInput3.......................................................................................................... 183
6.4. UseInput4.......................................................................................................... 183
6.5. PurchaseLivestock............................................................................................ 185
6.6. SellWeanlings................................................................................................... 187
6.7. SellOrphans....................................................................................................... 189
6.8. SellCullCows.................................................................................................... 189
6.9. SellStockerCalves............................................................................................ 190
6.10. BudgetReport.................................................................................................. 192
6.11. ResetMarketVariables.................................................................................... 193
Literature Cited......................................................................................................... 195

CHAPTER 7. Testing The Virginia Beef Cattle Simulation Model...................... 207
  Abstract............................................................................................................... 209
7.1. Introduction....................................................................................................... 209
7.2. Materials & Methods....................................................................................... 210
          7.2.1. Herd & Management Inputs................................................................. 211

xii
8.2. Summary.......................................................................................................................... 251
8.3. Implications...................................................................................................................... 254
Literature Cited....................................................................................................................... 255
APPENDIX A............................................................................................................................ 257
APPENDIX B............................................................................................................................ 274
APPENDIX C............................................................................................................................ 371
Vita......................................................................................................................................... 975
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3-1.</td>
<td>The General Layout Of The Virginia Beef Cattle Simulation Model</td>
<td>58</td>
</tr>
<tr>
<td>Figure 3-2.</td>
<td>A Diagrammatic Representation Of The Virginia Beef Cattle Simulation Model</td>
<td>59</td>
</tr>
<tr>
<td>Figure 3-3.</td>
<td>The Virginia Beef Cattle Simulation Model Function Diagram</td>
<td>60</td>
</tr>
<tr>
<td>Figure 5-1.</td>
<td>Lactation Curves Based Upon The Peak Week Of Lactation For Lactating Hereford, Angus, and Gelbvieh Cows In The Virginia Beef Cattle Simulation Model (NRC, 1996; Jenkins &amp; Ferrell, 1992; MARC, 1990)</td>
<td>173</td>
</tr>
<tr>
<td>Figure 6-1.</td>
<td>VBCSM Projected Cow-Calf Income Statement</td>
<td>198</td>
</tr>
<tr>
<td>Figure 6-2.</td>
<td>VBCSM Annual Projected Cow-Calf Statistics</td>
<td>199</td>
</tr>
<tr>
<td>Figure 6-3.</td>
<td>VBCSM Projected Partial Budget For the Stocker Calf Operation</td>
<td>202</td>
</tr>
<tr>
<td>Figure 6-4.</td>
<td>VBCSM Annual Projected Stocker Calf Statistics</td>
<td>203</td>
</tr>
<tr>
<td>Figure 7-1a.</td>
<td>Calving Season x Length Of Breeding Season Effects On Weanling Steer</td>
<td></td>
</tr>
</tbody>
</table>
and Heifer Income……………………………………………………… 232

Figure 7-1b. Calving Season x Length Of Breeding Season Effects On Cow-Calf Net Income, Stocker Calf Net Income, and Total Net Income…………………………………………………………………………………… 233

Figure 7-2. Calving Season x Culling Policy Effects On Heifer Income, Stocker Heifer Income, Cow-Calf Net Income and Stocker Net Income…………………………………… 236

Figure 7-3a. The Influence Upon Cow-Calf Net Income From The Interaction Between Length Of The Breeding Season x Culling Policy………………………………………………… 240

Figure 7-3b. The Influence Upon Culling Rate From The Interaction Between Length Of Breeding Season x Culling Policy............ 241

Figure 7-4. A Response Surface Displaying Predicted Cow-Calf Net Income From All Possible Management Combinations Of Calving Season, Length Of Breeding Season, And Culling Policy…………………………………… 242

Figure 7-5. A Response Surface Displaying Predicted Stocker Calf Net Income From All Possible Management Combinations of Calving Season, Length Of Breeding Season, And
Culling Policy

Figure 7-6. A Response Surface Displaying Predicted Total Net Income From All Possible Management Combinations Of Calving Season, Length Of Breeding Season, And Culling Policy

Figure A-1. The Introduction Page Of The Virginia Beef Cattle Simulation Model

Figure A-2. The Main Menu Of The Virginia Beef Cattle Simulation Model

Figure A-3. The First Dialog Page Of “Input Herd Values” Prompt Of The Main Menu Page In The Virginia Beef Cattle Simulation Model

Figure A-4. The Second Dialog Page Of The “Input Herd Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model

Figure A-5. The Third Dialog Page Of The “Input Herd Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model

Figure A-6. The Fourth Dialog Page Of The “Input Herd Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model
Figure A-7. The Fifth Dialog Page Of The “Input Herd Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model………………………………………………………… 264

Figure A-8. The First Dialog Page Of The “Input Forage Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model………………………………………………………… 265

Figure A-9. The Second Dialog Page Of The “Input Forage Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model………………………………………………………… 266

Figure A-10. The Third Dialog Page Of The “Input Forage Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model………………………………………………………… 267

Figure A-11. The Fourth Dialog Page Of The “Input Forage Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model………………………………………………………… 268

Figure A-12. The First Dialog Page Of The “Input Nutrition Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model………………………………………………………… 269

Figure A-13. The Second Dialog Page Of The “Input Nutrition Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model………………………………………………………… 270
Menu Of The Virginia Beef Cattle Simulation Model.......................................................... 270

Figure A-14. The First Dialog Page Of The “Input Market Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.......................................................... 271

Figure A-15. The Second Dialog Page Of The “Input Market Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.......................................................... 272

Figure A-16. The “Continue” Prompt That Is Displayed After The Presentation Of The YEAR No. 1 And No. 2 Financial And Herd Summaries Of The Virginia Beef Cattle Simulation Model.......................................................... 273
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2-1. A Comparison Among Beef Cattle Bio-Economic Simulation Models</td>
<td>41</td>
</tr>
<tr>
<td>Table 3-1. Nutritional Values Of Forage Species Simulated In The Virginia Beef Cattle Simulation Model</td>
<td>61</td>
</tr>
<tr>
<td>Table 3-2. Growth Rates In Pounds Per Acre Per Day For Forage Species In The Virginia Beef Cattle Simulation Model</td>
<td>62</td>
</tr>
<tr>
<td>Table 3-3. Nutritional Values Of Feed Supplement Ingredients In The Virginia Beef Cattle Simulation Model</td>
<td>63</td>
</tr>
<tr>
<td>Table 3-4. Descriptive Statistics For Several Traits In The Cattle Breeds Used In The Virginia Beef Cattle Simulation Model</td>
<td>64</td>
</tr>
<tr>
<td>Table 3-5. Alternative Culling Policies Simulated In The Virginia Beef Cattle Simulation Model</td>
<td>65</td>
</tr>
<tr>
<td>Table 3-6. Multiplicative Price Adjustments Used To Calculate The Price Of Different Grades And Weight Classes Of Weaned Cattle In The Virginia Beef Cattle Simulation Model</td>
<td>66</td>
</tr>
</tbody>
</table>
Table 3-7. Multiplicative Price Adjustments Used To Calculate The Price Of Different Grades And Weight Classes Of Stocker Cattle In The Virginia Beef Cattle Simulation Model………………………………………………………… 67

Table 3-8. The Monthly Pricing Index Used To Calculate The Price Of Cattle In The Virginia Beef Cattle Simulation Model… 68

Table 3-9. Multiplicative Price Adjustments Used To Calculate The Selling Price Of Cattle In The Virginia Beef Cattle Simulation Model………………………………………………………… 69

Table 4-1. Average slaughter weights and their standard deviations for 12 cattle breeds common to Virginia………………………………… 118

Table 4-2. Variables included in the Dystocia Function……………………………………………………………………………… 119

Table 4-3. The probability of death occurring On any day based upon age of the beef animal……………………………………………………………………… 120

Table 5-1. Averages And Standard Deviations For Monthly Rainfall In Virginia By Region For Years 1961 – 1991…………………………………………… 159

Table 5-2. The Multiplicative Effect Of Rainfall Upon Forage Growth In The Virginia Beef Cattle Simulation Model……………………………………… 160
Table 5-3.  Multiplicative Effect Of Weed
Infestation In The Virginia Beef
Cattle Simulation Model.............................................. 161

Table 5-4.  Multiplicative Effect Of Pasture
Maintenance In The Virginia Beef
Cattle Simulation Model.............................................. 162

Table 5-5.  Multiplicative Effect Of Pasture
Maintenance In The Virginia Beef
Cattle Simulation Model.............................................. 163

Table 5-6.  Multiplicative Effect Of Soil Erosion
In The Virginia Beef Cattle Simulation
Model................................................................. 164

Table 5-7.  Monthly Temperature Average And
Variation In Virginia By Region
For Years 1961 –1991 In Degrees
Fahrenheit.............................................................. 165

Table 5-8.  Multiplicative Effects Of Using
Or Not Using Anabolic Implants On
Determining The Amount Of Dry Matter
Intake In The Virginia Beef Cattle
Simulation Model..................................................... 166

Table 5-9.  Multiplicative Effects Of Temperature
On Determining The Amount Of Dry
Matter Intake In The Virginia Beef
Cattle Simulation Model.............................................. 167

Table 5-10.  Multiplicative Empty Body Fat Effects
On Determining the Amount of Dry Matter Intake In The Virginia Beef Cattle Simulation Model........................................................................................................... 168

Table 5-11. Multiplicative Modifiers Used To Adjust The Proportion Of The Animal Diet Furnished By Supplemental Feeds Relative To The Initial Daily Forage Mass Available (Kgs/DM/Cow) In The Virginia Beef Cattle Simulation Model... 169

Table 5-12. The Effect Of Shrunk Body Weight On Body Condition Score And Net Energy For Maintenance.......................................................................................................................... 170

Table 5-13. Age Effects Upon Milk Yield In The Virginia Beef Cattle Simulation Model... 171

Table 5-14. The Peak Week Of Lactation And The Milk Energy Multiplier Used To Adjust The Energy Value To The Estimated Volume Of Milk Produced By Each Breed In The Virginia Beef Cattle Simulation Model.......................................................................................................................... 172

Table 5-15. Internal Insulation Factors Used In The Virginia Beef Cattle Simulation........ 174

Table 5-16. The Effect Of Panting Caused By Heat Stress On The Daily Dry Matter Intake Required To Meet Beef Cattle Maintenance Requirements.......................................................... 175
Table 6-1. The Amount Of Shrinkage For Cattle By The Marketing Method Used In The Virginia Beef Cattle Simulation Model... 204

Table 6-2. Auction and Virginia Tel-O-Market Fees used in the Virginia Beef Cattle Simulation Model.............................. 205

Table 6-3. Multiplicative Price Adjustments Used To Calculate The Price Received For Culled Cows In The Virginia Beef Cattle Simulation Model........................................ 206

Table 7-1. Important Calving Season Effects.............. 245

Table 7-2. Important Length Of Breeding Season Effects.......................................................... 246

Table 7-3. Important Culling Policy Effects............. 247

Table 7-4. Important Year Effects On Selected Incomes and Traits By Years 1 - 3................. 248

Table 7-5. Descriptive Statistics Of Selected Biological Traits By Calving Season........ 249
Chapter 1. Introduction

1.1. General Background

1.2. Objectives Of The Study

Literature Cited
Chapter 1

Introduction

1.1. General Background

Cow-calf herds have traditionally been a mainstay of the cattle industry in the South (Kolajo & Martin, 1994). They are one of the most important agriculture pursuits in Virginia. In Virginia, cattle and calves ranked second in cash receipts for all agricultural commodities in 1997 while Virginia ranked twenty fourth among cattle producing states in the nation (VASS, 1997). In 1996, Virginia had 2.89% of the nation’s total beef cow herds and 2.16% of the total number of beef cows in the United States (Cattle-Fax, 1997). Additionally, Virginia ranked fourteenth in the nation in feeder cattle numbers in 1997. Virginia's abundant grasslands and part-time farmers give the industry a strong competitive advantage compared to other beef producing states (Purcell, 1993).

Virginia’s competitive position in this vital state industry is maintained through the support of many dynamic and innovative marketing programs sponsored by the Virginia Cooperative Extension Service in cooperation with state beef cattle organizations such as the Virginia Cattlemen's Association (Purcell, 1993). Since the beef cattle industry is less vertically integrated than other food producing
activities, profitability in the cow-calf sector depends primarily upon low cost grazing while the finishing of feeder calves depends upon the use of feed grains or silage (Miller et al., 1980). Profitability during the stocker period, however, depends not only upon the price of feed grains and the demand for beef but additionally upon the geographic region of the country in which cattle are grown.

Over the years, beef cattle producers have sold their cattle at terminal markets, local auction markets, and though dealers, as well as directly to feedlots, meat packers, and other stock farmers. Because most of the their time is devoted to production, cow-calf producers generally are willing but inexperienced part-time participants in marketing programs (Lessor, 1993). Many cow-calf producers fail to distinguish the difference between selling, a dynamic, producer driven action, and marketing, a consumer driven activity. From a marketing viewpoint, marketing is equal to production in importance, and production must thus be attuned to the market. Cattle producers should devise a marketing strategy that uses a systematic approach to decision making aided by a variety of tools possibly including complex computer aids as well as simple logic.

A modern producer must know the advantages and disadvantages of available marketing alternatives because traditional alternatives may or may not be optimal (Lessor,
1993). Cattle producers that are interested in achieving the highest net farm income should consider not only their production strategies but also, prior to the time that the animals are sold, price and market alternatives that are available to them. Often, however, they do not consider marketing until their animals are sold, and it is then too late to do anything other than to accept the highest bid (Davis, 1988).

While producers may have different goals in marketing livestock, their economic objective is usually to maximize net return. Since the application of science to agriculture involves managing biological resources to achieve economic objectives, it is theoretically possible to describe these systems in mathematical models. Various constraints can then be changed to predict the biological or economic impact (Forbes & Oltjen, 1986). The development of computers and programming aids now enables modelers to construct mathematical simulations that describe dynamic biological events in a manner that was impossible to visualize only a few years ago. These simulations may identify optimal plans, but in addition, they can provide quantitative information to farm managers without being designed or modified to fit specific situations (Musser & Tew, 1984). These programs can identify interactions among investment, production and marketing strategies when risk is present
(Johnson & Boehlje, 1983). Properly interpreted simulation output can soothe the frustrations of farm managers as well as the economists that advise them. General recommendations can be obtained at the farm level from flexible, realistic computer simulations that allow plans to be revised either as new information or the results of previous decisions become available (Trapp & Walker, 1986). Simulations can be programmed to use deductive, short-term, tentative information exactly as it becomes available in real time in addition to assigning probability values to various outcomes so that alternatives can be compared. As the farm manager gains experience using a simulation program, he will be able to get faster answers to his questions.

In designing the computer simulation program, a measuring system that will compare alternative livestock production systems can be mathematically modeled. Although livestock productivity can be expressed in several ways, the variables chosen should reflect economic efficiency to permit the allocation of scarce resources when choosing among production alternatives (Upton, 1989). Valid comparisons are then made using performance measures that should be descriptive, analytical, or prescriptive in order to identify the strengths or weaknesses of alternative production and marketing strategies (Peacock, 1987). Most simulation models use cash flow to evaluate investment alternatives (Chudleigh & Cezar, 1982).
1.2. Objectives Of The Study

The objectives of this study are to:

1. Develop a computer simulation model that will simulate the annual production of a Virginia cow-calf firm, including modules for forage production, nutrition, reproduction and life events, and marketing.

2. Use the model to analyze the effect upon profitability of Virginia beef cattle production firms from changes in production and marketing strategies.
LITERATURE CITED


Trapp, James N. and Odell L. Walker. 1986. Biological
simulation and its role in economic analysis. In: 


CHAPTER 2. Literature Review

2.1. Introduction

2.2. Introduction to Operations Research

2.3. Simulation in Beef Cattle Production

2.4. Deterministic vs Stochastic Simulation Models

2.5. Dynamic vs Static Simulation Models

2.6. Risk Management in Simulation Modeling

2.7. C++ & Object-Oriented Programming

2.8. A Program's Lifecycle

2.9. Debugging or Verifying Simulation Models

2.10. Validating Simulation Models

2.11. Beef Cattle Simulation Models

   2.11.1. Colorado Model

   2.11.2. HERDPLAN

   2.11.3. Kentucky Beef-Forage Model

   2.11.4. Notter Model

   2.11.5. Spur Model

   2.11.6. Texas A&M Simulation Model

2.12. Conclusion

   Literature Cited
Chapter 2

LITERATURE REVIEW

Abstract: Operations research is a scientific approach to decision making that uses quantitative models to analyze and predict the behavior of systems influenced by human decisions. The scientific approach uses a mathematical or logical model to describe the system and its most important features. Such models can be used advantageously in production agriculture, but managers should be aware of a model’s limitations. A simulation is used to imitate real world processes to predict various outcomes. Bio-economic simulations mathematically simulate both biological systems and the ensuing economic consequences to detect system deficiencies and make improvements. No computer model can completely simulate the complicated and intricate risk interactions that confront beef cattle producers daily. C++ is a high level programming language designed to use object-oriented programming that breaks a problem into self-contained, interrelated objects for manipulation and abstraction. If a model is "valid", then conclusions reached from it should be similar to those that would be reached after experimenting with the actual system. Several beef cattle simulation models have been developed. These include the Colorado State Beef Production Model, HERDPLAN, the Kentucky Beef-Forage Model, the Notter Model, SPUR, and the Texas A&M Beef Cattle Simulation Model.

2.1. Introduction

The purpose of this chapter is to provide a background in the use of computer programming and operations research in beef cattle management. This will be accomplished through a review of operations research, the use of simulation programs in beef cattle production, types and characteristics of simulation programs, and the object-
oriented C++ programming language.

2.2. Introduction to Operations Research

Operations research is a scientific approach to decision making that uses quantitative models to analyze and predict the behavior of systems that are influenced by human decisions (Ecker & Kupferschmid, 1988; Hillier & Lieberman, 1986). It is applied to problems that are concerned with conducting and coordinating the operations or activities within an organization (Hillier & Lieberman, 1986). A scientific approach implies the use of a systematic methodology in which a mathematical or logical model of the problem is created and applied to help managers make better decisions (Ecker & Kupferschmid, 1988). Since the model in and of itself is an abstraction of a system, it defines the system being modeled by representing its most important features. Operations research involves research into the fundamental properties of a system in an attempt to determine the most optimal solution to a problem consistent with the objectives of an organization (Hillier & Lieberman, 1986). By using a mathematical or logical model, the problem is precisely described by the simultaneous evaluation of its interrelated properties to arrive at a reliable solution that meets the constraints of the model. Models, however, depend upon the validity of their assumptions (Ecker & Kupferschmid, 1988).
An operations research approach to management, although not often used in production agriculture, can be a beneficial tool. This approach allows a system to be modeled in order to provide computer solutions to future problems. This permits more management time to be allocated to decision-making (Ecker & Kupferschmid, 1988). However, it forces managers to be more explicit about their objectives, assumptions, and identification of constraints. Most importantly, decision-making managers can completely review a situation, change the conditions under which decisions are made, and examine the effects of those changes on the system without serious damage or excessive cost.

Sometimes, however, managers become so enamored with a model that they forget that with its simplifying assumptions, it does not truly represent the real world in which their decisions must be made. Often managers are unaware of the limitations of the model, including the fact that it is only a decision support system whose solutions must be combined with experienced judgement and discerning intuition to be effective.

2.3. Simulation in Beef Cattle Production

A simulation is a technique for using computer programs to imitate real-world processes (Law & Kelton, 1991). The process of interest is called a system and to study it scientifically, we must make a set of assumptions in the form
of a logical or mathematical model that is then used to gain some understanding about how the corresponding system behaves. Beef production involves the management of biological systems to economically produce a consumer product (Forbes & Oltjen, 1986). It is at least theoretically possible to encode mathematical models into computer software programs that can predict various outcomes with some degree of confidence as input values are varied. In biological simulation, most models are dynamic models that evolve over time and deterministic (non-random) in nature, although some are stochastic or random models (Dougherty et al., 1985). In cattle production, simulation models are used for classroom teaching tools, management tools, ration formulation, performance appraisal, and animal breeding programs by both educators and industry (Fox & Black, 1984).

Bio-economic computer simulation programs are models that mathematically simulate both biological systems and the ensuing economic consequences by measuring discounted cash flows to support investment decisions (Chudleigh & Cezar, 1982). Although they can be used to compare the performance of different systems, they must compare systems under standardized conditions if they are to be valid for detecting system deficiencies and making improvements in them (Upton, 1989). Accuracy is essential for useful simulations, but faulty applications of theory will result in
simulations that are incapable of dealing with beef production and beef marketing systems that differ across time and space (Denham & Spreen, 1986). For this reason, even though simulations are useful in making plans, their results should not be accepted without question. In fact, simulation programs can never identify the optimum strategy for any cattle producer; they can only provide quantitative comparisons (Musser & Tew, 1984). Normally, more general simulation models will have greater flexibility (Chudleigh & Cezar, 1982). The failure to model beef cattle systems as intact operating units that utilize integrated regulatory processes to simulate reproduction, growth, and death will result in a simulation model that gives inappropriate responses to input changes (Denham & Spreen, 1986). An effective bio-economic model, however, can be used to make valuable economic projections.

Expected market prices are often used to make financial projections because profit is a rational objective for the cattle producer (Upton, 1989). In an effective beef production and marketing model, cattle numbers for different classes are calculated at any point in time using variables such as birth rates, age at first calving, death rates, buying and selling policies, and culling procedures (Chudleigh & Cezar, 1982). Comparisons between different classes of cattle must be calculated consistently in terms of a standard time period and the type of animal-units that
are used (Upton, 1989). A bio-economic simulation model will use information in the same way that it becomes available in the "real world" which is in a tentative, deductive, piecemeal manner that may only be valid in the short term (Trapp & Walker, 1986). This means programmers and program users should rely upon their own judgement to interpret simulated predictions exactly as cattle producers must rely on their judgement to make production and marketing plans (Upton, 1989). Although cattlemen have been slow to accept computers and bio-economic simulation software, its acceptance rate has not been significantly different from the acceptance of other technology in other agricultural sectors (Tomaszewski & Fourdraine, 1992).

Beef cattle simulation models can be flexible management and research tools. They can contain both stochastic and deterministic subroutines. Their objective may be to identify and evaluate the strengths and weaknesses of different production and marketing systems. When encoded and validated, beef cattle simulation systems that can summarize production data may be quickly put to use by progressive managers (Crandall, 1992). A well designed livestock simulation system will first develop concepts for important life events such as, for example, the number of calves that are born within a specific period of time. It next implements the actions necessary to achieve these abstractions such as, for example, defining the criteria
that allow breeding age females to become pregnant and maintain pregnancy. Then, it performs regularly scheduled performance evaluations. When the goals are not met, the model will revise the goal and/or the action necessary to a more realistic level before repeating the sequence.

Since cattle operations involve the use of a set of conceptual and practical functions that interact to affect productivity and profitability, the simulation must furnish information about the management of available resources (Balliet, 1977). If well designed, a simulation model can: investigate the age or growth stage at which to sell cattle; investigate whether or not to purchase or retain heifers to replace culled cows; investigate the most opportune culling schedules for breeding cows; investigate the impact of changes in the breeding season; investigate which marketing strategy will yield the most income; and compare different alternatives in terms of a cash flow analysis that can be constrained in various ways (Chudleigh & Cezar, 1982).

The first step in designing a beef cattle simulation program is to describe all biological and management factors known to importantly influence cattle production (Fox, 1992). For example, factors that should be considered if predicting growth are weight, frame size, breed, sex, growth stimulants, environmental stress, management systems, and the animal's physical and thermodynamic ability to consume...
Because the efficiency of beef production is constrained by the annual calf crop, the simulation model must address reproductive performance (Denham et al., 1991). Although reproductive performance has been modeled in the past, the simulations have not been used to investigate the management tactics that producers need in this volatile era (Hohenboken, 1992). Reproductive events should be modeled as the stochastic processes that they are (Azzam et al., 1990). The type of distribution used by the simulation can provide more understanding of the reproductive process. The primary reproductive events are conception and parturition.

This means that herd performance averages should be used to simulate reproduction in order to have a continuous distribution for those events (Denham et al., 1991). Gestation should be linked to a variable time span for both the breeding season and the postpartum return to estrus in order that mathematical uncertainty about its success can be incorporated. Finally, because reproduction is ultimately related to financial success, the simulation should reflect saleable calf production in a stochastic manner, reflecting variance in the weight and grade of weaned calves. Such a subroutine could show that managerial decisions should be based not only upon biological responses but, even more importantly, upon economic decisions (Johnson & Notter,
The simulation model should also reflect forage availability, because the decision to retain weaned calves will depend upon the availability of forage (Angirasa et al., 1981). Any competition between cows and calves for forage depends upon the length of time that the calves are retained. The simulation should dynamically model plant and animal growth so that the interaction between forage quality, forage quantity, and animal intake can be incorporated (Loewer et al., 1987). The maximum rate at which forage may be eaten by a ruminant animal will be jointly determined by the forage's physiological state and its availability. Dry matter availability will limit forage intake independent of its quality.

2.4. Deterministic vs Stochastic Simulation Models

A deterministic simulation is a simulation that does not have any probabilistic or random components. Output is uniquely determined by the model's input relationships and quantities. Since in the "real world," the variances as well as the means of input variables influence outputs, it is dangerous to ignore randomness (Law & Kelton, 1991). Stochastic simulation models recognize that most real world systems exhibit some sort of random behavior by randomly sampling values from a defined distribution of stochastic inputs used in modeling a system. By using such inputs,
stochastic simulation models produce outputs that follow some random distribution themselves and therefore must be treated only as estimates of the true outcomes of the system.

2.5. Dynamic vs Static Simulation Models

A static simulation model is a representation of a system at a particular point in time while a dynamic simulation model represents a system as it evolves over time, i.e., a system in which time is an important element (Law & Kelton, 1991). Most dynamic models are discrete event simulations in which variables change at various times as the model attempts to mimic the real-world process.

2.6. Risk Management in Simulation Modeling

No computer model can completely simulate complicated and intricate risk interactions confronting beef cattle producers each day during the production process. Risk is the possibility that different outcomes can result from any decision. Risk probabilities can guide decision making (Johnson & Boehlje, 1983). Risk can be evaluated by assigning probabilities of occurrence to different random events and then selecting the strategy that has the highest expectation of occurring. It may be as important to measure a venture's risk as it is to measure its expected return (Simpson et al., 1977). If producers are actually concerned about risk, then probability distributions for different
levels of risk are required in order to model the proper distribution of risk (Trapp & Walker, 1986).

One useful technique in simulating risk is sensitivity analysis (Simpson et al., 1977). This technique measures the variance of net present values resulting from changes in a system’s variables. The net present value is calculated by keeping all values constant except the variable being examined. The different variables are ranked by the effect that they have on changes in net present value.

Cattle ventures require more attention to risk than other business ventures because of their inherent unpredictability due to exogenous factors such as weather, disease, pests, changes in the price of inputs and outputs, and non-agricultural factors (Berbel, 1993). However, cattle production is a rational enterprise for moderately risk averse cattle producers because hedging strategies can be used to protect expected returns (Johnson & Boehlje, 1983). A good marketing strategy makes the beef cattle producer more able to manage risk. However, even in addressing risk, simulation models can only provide quantitative information that may be used as an aid to solve a problem (Musser & Tew, 1984).

2.7. **C++ & Object-Oriented Programming**

A high level computer programming language is a series of rules, symbols, and special words used to construct a
software program that can be stored, retrieved, and processed on a computer (Dale et al., 1996). The C++ programming language is a high level programming language that was developed by A T & T's Bell Labs in 1985 to add data abstraction and object-oriented programming to its earlier C programming language. The ++ signifies an enhanced or incremented version of the C language. The phase "high level programming language" indicates that C++ uses a compiler to translate its instructions into a binary coded machine language used directly by the computer.

All computer programs are implementations of algorithms that are step by step instructions for solving a problem in a measurable amount of time (Dale et al., 1996). Like other programming languages, C++ uses four basic structures to initiate action: the sequence of the program code, the selection of the code that will be acted upon, looping, and its subprograms. In C++, all subprograms are referred to as functions. Each C++ computer program contains one or more such functions. However, C++ is not a structured or procedural type of programming that only consists of a collection of interacting functions. Rather it is an object-oriented programming language that allows a problem to be broken into self-contained interacting objects that can be manipulated while ignoring the unessential parts of the system (Farrell, 1998). Variables and classes of one object can be related to another object in several ways.
(Standish, 1995). This allows not only the object to be analyzed but also any other routine that acts on that object (Farrell, 1998). Because current objects can share the properties of previous objects, not only is the time required to create a new object reduced but the object can be programmed to work in a different manner in other objects. Thus, a framework is provided in which components can be expressed and reused as needed (Standish, 1995).

Since object-oriented programming can be used to express a system's structure in modules rather than in a step by step procedural manner, the program may have an enhanced capacity to tolerate change and modification (Standish, 1995; Farrell, 1998). This reduces the cost of maintaining and modifying the program. Desired changes can be directly programmed into local functions instead of making the many non-local text changes that are required when using traditional programming techniques. The trade-off with object-oriented programming is however reduced efficiency because the search time to identify and allocated suitable blocks of storage memory is significantly increased (Standish, 1995). This is usually not a problem on new generations of computers.

2.8. A Program’s Lifespan

A computer program's lifespan is the period of time that begins with the initial conception of the system to its
eventual retirement from service (Standish, 1995). The basic lifespan of a program consists of three phases: (1) the problem-solving phase, the implementation phase, and the maintenance phase (Dale et al., 1996). In the problem-solving phase, the problem and its solution are defined and an algorithm is developed and verified. In the implementation phase, the algorithm is translated into a high level programming language and its instructions are manually checked in order to find errors, determine the source of the errors, and make corrections. In the maintenance phase, the program is used and modified to meet changing requirements and to correct any further errors discovered while using it. The desired end result is a cheap, reliable, safe, efficient computer software program that does what it was designed to do (Standish, 1995).

2.9. Debugging or Verifying Simulation Models

Verification (or debugging) is determining that a simulation or computer program performs as intended. It involves checking the translation of the conceptual simulation model, its flow charts and assumptions into a properly working simulation program (Law & Kelton, 1991). When a simulation or computer program does not do what it should do, some type of logical error exists and must be fixed (Dale et al., 1996). This process is repeated until the program does what it is supposed to do.
There are two methods used to verify or debug a model (Schmisseur, 1992). The first method independently tests each module of the model and then merges the modules into the working model without further testing. The second method uses two strategies. The first strategy is called the "top-down" method because it tests the most important module of the model first and then proceeds to test any lower level modules (Schmisseur, 1992; Dale & Weems, 1994). The second strategy is called the "bottom-up" method because it examines the lowest level module prior to proceeding to more important modules (Schmisseur, 1992). "Bottom-up" is of limited applicability because a working model is not available until all testing is complete.

Each method can verify the model in two ways (Schmisseur, 1992). In the first way, a static analysis is used to survey the system's requirements, the documents of design, and the program's code without actually doing any executions of the code. It is very effective in discovering logic and coding errors that were inserted as the model was developed. The second method uses a dynamic analysis that executes every statement in the simulation program, uses every path in the program, and uses each condition in the decision statements at least once. A well-planned verification program in conjunction with a well-planned validation system can inspire greater confidence in the simulation model.
2.10. Validating Simulation Models

There are no hard and fast rules dictating the procedures to be used to validate a simulation model (Baldwin, 1992). The best method is to use the simulation model to simulate the behavior of an existing system or enterprise. If the results are in close agreement with reality, the model can be considered valid, and confidence in it will be established. If not, more work is needed before the model can be trusted. The data used to test the model, however, must be carefully selected in order to give the model a rigorous examination and to identify its general operating characteristics (Thornton, 1985).

Since a simulation model is a computer coded flow chart of a set of mathematical equations, the model must be a good approximation of an actual system (Sorensen, 1990). Validation consists of three components: rationalism, empiricism, and positive economics (Naylor & Finger, 1967). Rationalism means that the model is based upon logical inferences whose assumptions cannot necessarily be empirically validated. In this context, empiricism means that each assumption in the model should be independently validated by comparison against actual observations. Positive economics means that the model should be judged on its ability to predict actual outcomes in order to determine whether or not the conceptual model is an accurate representation of the real-world
system. A "goodness-of-fit" test may determine the degree of conformity between simulation results and observed system outcomes using identical input values for both (Sorensen, 1990). If the model is "valid", then decisions made with it should be similar to those that would be made after experimenting with the actual system (Law & Kelton, 1991).

Another valuable evaluation tool is field-testing. Field-testing uses the simulation model to provide recommendations in actual environments under actual conditions (Harrison, 1991). It puts the responsibility of testing the simulation model upon the shoulders of the cattle producers who will eventually use the model in a commercial context and who may not have much computer training. This method can uncover deficiencies in a simulation model very quickly. The system should be refined until an acceptable standard of performance is obtained.

2.11. Beef Cattle Simulation Models

Several simulation models currently in use are compared in Table 2-1. Most were designed for research purposes. Although the Kentucky Beef-Forage Model has been described as both a research model and a decision-support aid, it may be too complicated for the average beef cattle producer to use.

2.11.1. Colorado Model

The Colorado State University Beef Production Model
(Bourdon, 1998; Bourdon & Brinks, 1987a, b, c) is a modified version of the Texas A&M Beef Production Model (Sanders & Cartwright, 1979). It contains most of the modifications of Notter (1979a, b, c), especially those that relate to nutrient intake in the feedlot. It is a dynamic, stochastic, beef cattle research model that operates at the level of the individual animal (Bourdon, 1998; Hanson et al., 1993). The model is designed to reflect a typical northern plains, range cattle environment that features a semi-arid shortgrass steppe with relatively cold winters and temperate summers (Bourdon & Brinks, 1987a). Biological routines simulate growth, fertility, pregnancy, calving, and death and allow 14 traits to be studied (Hanson et al., 1993).

An economic module was added to the program that can analyze the results from the biological model (Bourdon & Brinks, 1987a). Costs are categorized as being fixed to the farm unit and not to the cattle, non-feed costs that are variable to the farm unit but fixed to the number of cattle, and feed costs. No limit on capital is assumed and interest costs are calculated on monthly cash flows. Revenues are calculated from a fixed price per pound received for weanling calves, fed calves, and culled cows.

The Colorado model assigns the entire herd the same genetic potential (Bourdon & Brinks, 1987a). Additionally, heterosis is set at 75% of its maximum potential, a level
that Colorado researchers considered typical of many commercial herds within the state. Also, no specific mating systems are simulated.

2.11.2. HERDPLAN

HERDPLAN was developed as a decision-support tool for beef cattle producers. Although it was programmed and tested in Nigeria, it is designed for worldwide use. It is composed of four modules: environmental, physical, accounting, and price (Oguntade, 1991). The environmental component simulates the interactions between rainfall, pasture yield, and stocking density. The physical component simulates the interrelationship between variables that are used to describe the cattle. The accounting module interacts with the pricing function in order to prepare balance sheets, income statements, and herd reports. The rate of change in prices over time is user-specified. Also, in order to run the simulation, the user must specify the weaning age and growth rates of the calves by breed, the ages of all simulated cattle, the forage yields and digestibility’s, initial values of all capital assets, and such system parameters as calving percentages and mortality rates. The program then provides management reports at the end of each simulation year. HERDPLAN can be used to compare alternative production strategies of extensive beef-cattle systems that vary in their costs.
2.11.3. Kentucky Beef-Forage Model

The Kentucky Beef-Forage Model is a combination continuous-discrete event, deterministic, beef cattle herd simulation research model programmed in FORTRAN that uses the GASP IV simulation language as described by Pritsker in 1974 (Loewer & Smith, 1986). Time step is daily. This model is a result of the interdisciplinary efforts of animal scientists, agronomists, agriculture engineers, and agricultural economists from 25 states. It is a total farm systems program that includes a series of interconnected models: BEEF, GROWIT (version 2), BABYBEEF, BEEF-S156, SAM (Surface Area Model), BEEFEM, and GRAZE. These models allow the outcome of a management decision upon plant and animal production, energy consumption, or an economic return to be evaluated.

The series of models originated from BEEF. BEEF is composed of four sections: plant, animal, energy, and economic. It has no direct optimization features but is rather a “consequence of actions” model that is time stepped daily (Loewer & Smith, 1986). It groups animals into different categories according to their age, sex, and reproductive status to simulate their growth and reproduction. An animal may change categories as a result of changes in age, breeding status, castration, birth, death, or movement between fields based upon internal
accounting and input specifications. Output is reported through income statements, energy statements, bank accounts, note schedules, and purchase schedules.

Plant growth is simulated in the GROWIT (version 2) model by utilizing temperature, rainfall, and soil fertility (Loewer & Smith, 1986). Grazing pressure is defined by incorporating logic concerning both the plant cell content and the cell wall with the plant’s physiological age to define its potential digestibility. The quality factors defined in GROWIT (version 2) are interfaced with the grazing patterns of the cattle to determine the dry matter intake of the cattle.

BABYBEEF relates NRC growth equations with environmental factors and physiological growth, while BEEF-S156 uses physiological growth as expressed by changes in body composition to alter feed intake (Loewer & Smith, 1986). Interior body temperature is calculated by using both BEEFEM and SAM (Surface Area Model). SAM uses geometric shapes to describe the heat flows to and from the animal. The digestibility of the beef animal’s dry matter intake is maximized using GRAZE which interfaces selective grazing logic with plant growth logic from GROWIT (version 2) and animal growth from BEEF-S156. This allows the animal to select plant material from a variety of sub-areas within the total grazing area.
2.11.4. Notter Model

The Notter model is a dynamic, deterministic, beef cattle herd production, research simulation model (Notter et al., 1979a, b, c). It is a modification of the original version of the Texas A&M Beef Cattle Simulation Model (Notter, 1997). Notter and his associates modified the TAMU model in order to study the effects of milk production, body size, and crossbreeding systems on the biological and economic efficiency of beef production in a Midwestern cow-calf-feedlot management system (Notter et al., 1979a, b, c). Because the equations in the TAMU model were derived using Hereford and Angus data, modifications had to be made to accommodate the slower relative fattening rate in large type cattle such as many of the Continental European breeds (Notter et al., 1979b). Other modifications included allowing the rates of conversion of metabolizable energy to net energy for maintenance, lactation, fat gain, and protein gain to vary with the digestibility of the feedstuffs. Additionally, dry matter availability was assumed not to constrain the animal’s dry matter intake, while the physiological limit on dry matter intake was assumed proportional to the animal’s metabolic body size (Notter et al., 1979a).

2.11.5. Spur Model

SPUR (Simulation of Production and Utilization of
Rangelands) is a stochastic, grassland-ecosystem, research and management simulation model that can simulate rangeland hydrology-plant-animal interactions. (Wight & Skiles, 1987; Hanson et al., 1988; Hanson et al., 1993). It is incremented on a monthly time clock. SPUR is made up of five modules: climate, hydrology, plant, animal (both domestic and wildlife), and economics (Wight & Skiles, 1987). The climate component operates outside the model, and data can be either stochastically generated or obtained from existing weather records. In the plant portion of the model, net photosynthesis is used to predict forage production. Up to 15 plant species growing on up to 36 sites can be simultaneously simulated (Hanson et al., 1993). However, there are two existing versions of SPUR: a field scale or grazing unit version and a basin scale version. The field scale version only simulates up to seven plant species on up to nine sites within a grazing unit (Wight & Skiles, 1987). In SPUR, the animal component includes both cattle and wildlife. The Colorado Beef Production Model is used to simulate cattle growth in SPUR (Bourdon, 1998; Hanson et al., 1993). Forage intake is computed using FORAGE, a deterministic model that interfaces both with the Colorado Beef Production Model and SPUR. Forage is a function of animal demand, animal weight, and the quantity and quality of the available forage for each time step. Wildlife
species including insects are considered as fixed consumers and allowed to have first access to the available forage (Wight & Skiles, 1987). Animal production or pounds of beef gain are used to estimate benefits and costs of alternative grazing systems, range improvements, and animal management. A unique feature of the animal component is that an allowance can be made for the selection of the grazing sites as well as the forage preferences of the herbivores being simulated by the user. This in turn will determine where the cattle will graze and how much of each forage that they will harvest (Wight & Skiles, 1987). Additionally, a forage’s properties will cause the animal to either ingest it or reject it.

2.11.6. Texas A&M Simulation Model

The Texas A&M (TAMU) Beef Cattle Simulation Model was originally programmed as a dynamic, deterministic, beef cattle herd production, research simulation model and called the Texas A&M Beef Cattle Production Model (Sanders & Cartwright, 1979). It was subsequently up-dated to contain stochastic elements for death, birth, estrus, conception, and removal. It was then modified and restructured to account for individual animals and renamed the Texas Beef Cattle Simulation Model (Baker, 1982; Doren et al., 1985; Cartwright & Doren, 1986). The TAMU model is programmed in FORTRAN and has a monthly fixed time advance that makes it
insensitive to changes in pasture growth (Whelan, 1984). It has been used to simulate beef cattle production under a variety of management schemes and environments with cattle that differ in genotype for size, growth, and milk production (Sanders & Cartwright, 1979). Herd dynamics place few limitations on herd size, management options, time and length of breeding season, and weaning polices although the user can impose limits by making necessary changes in the subroutines and input section of the main model. Animal performance level is simulated both from specified feed resources and beef cattle production potentials. Environmental conditions are described primarily by a feed supply that is characterized monthly by its crude protein content, dry matter digestibility, and availability. The values assigned to feed supplies influence system behavior but since they are set outside the system’s structure, they remain static in response to any changes generated in the system (Sanders & Cartwright, 1979). Thus, the quantity and quality of available pasture cannot respond to animal consumption and fails to mimic any effects that may result from modifications in the system (Whelan, 1984). Additionally, no stipulation is made for the effects that feed supplements have on pasture consumption. Therefore, the TAMU model may only be adequate for predicting response from an unlimited supply of feed.
2.12. Conclusion

Beef cattle production involves three stages: cow-calf, stocker-grower, and finishing cattle. In order to simulate any of these operations, the modeler must be familiar with beef cattle bio-economic systems. The modeler must also be reasonable concerning the utilization as well as the limitations of mathematically modeled systems. Finally, modelers must ensure that their simulation models are properly verified and validated. The simulation model should be able to compare production and marketing strategies in a standardized way that will let the cattle producer assess the performance and weigh the risks involved in alternative schemes so that scarce economic resources can be allocated in the most profitable manner.

Although several stochastic beef cattle research simulation models are currently in use, they have not been modified into decision-aid tools for beef cattle producers. In fact, the only research simulation model currently available for cattle producers to use as a decision-aid tool is the Kentucky Beef-Forage Model which is a complicated, deterministic "consequence of actions" model.
LITERATURE CITED


Bourdon, R. M. 1998. Personal communication.


Table 2-1. A Comparison Among Beef Cattle Bio-Economic Simulation Models.

<table>
<thead>
<tr>
<th>Simulation Model</th>
<th>Type</th>
<th>Animal Module</th>
<th>Forage Module</th>
<th>Time Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colorado Model</td>
<td>Stochastic Research</td>
<td>Individual animal</td>
<td>NO</td>
<td>Monthly</td>
</tr>
<tr>
<td>HERDPLAN</td>
<td>Deterministic Decision-Support</td>
<td>Herd</td>
<td>NO</td>
<td>Monthly</td>
</tr>
<tr>
<td>Kentucky Beef-Forage</td>
<td>Deterministic Research</td>
<td>Herd</td>
<td>YES</td>
<td>Daily</td>
</tr>
<tr>
<td>Notter Model</td>
<td>Deterministic Research</td>
<td>Herd</td>
<td>NO</td>
<td>Monthly</td>
</tr>
<tr>
<td>Spur</td>
<td>Stochastic Research</td>
<td>Individual animal (Colorado Model)</td>
<td>YES</td>
<td>Monthly</td>
</tr>
<tr>
<td>Texas A&amp;M Beef Cattle</td>
<td>Stochastic Research</td>
<td>Individual animal</td>
<td>NO</td>
<td>Monthly</td>
</tr>
</tbody>
</table>
Chapter 3 The Virginia Beef Cattle Simulation Model

3.1. Introduction

3.2. The Virginia Beef Cattle Simulation Model

3.2.1. Forage and Nutrition

3.2.2. Reproduction and Life Events

3.2.3. Marketing

3.2.4. Tools

LITERATURE CITED
Chapter 3

The Virginia Beef Cattle Simulation Model

Abstract: The Virginia Beef Cattle Simulation Model is a user friendly, dynamic, stochastic, decision-support, simulation model specifically adapted to Virginia. Its primary purpose is to serve as a decision-aid for Virginia cattlemen to evaluate beef cattle management options such as whether to retain weanling beef cattle through the stocker growth stage. It is composed of five primary source-code modules: forage, nutrition, reproduction and life events, marketing, and a tools module containing a function that assigns values to stochastic variables from appropriate distributions.

3.1. Introduction

The purpose of this chapter is to describe the Virginia Beef Cattle Simulation Model (VBCSM). This will be accomplished by providing an introduction to the VBCSM model that includes flow diagrams of the model and a short introduction to each of its five major modules.

A bio-economic simulation model is a mathematical description of a production system that can serve as a management decision tool (Denham & Spreen, 1986). A bio-economic beef cattle model should be able to predict animal growth and other biological functions as well as to estimate profit or loss given such biological and physical inputs such as the animals’ genetic potentials and the environment in which they are expressed. Figure 3-1 represents a
generalized schematic of the VBCSM.

Common concerns of Virginia beef cattle producers are:

How can profits be increased given constraints on capital?

Will retaining ownership of calves beyond weaning increase profits? If so, how long should stocker cattle be retained? Is retained ownership of calves always profitable?

What is the best culling policy for cows within a breeding herd?

How can the profitability of different breed or breed combination of beef cattle be ranked?

Is it more profitable to use a spring calving season, a fall calving season, or a combination of the two?

Should calves be sold direct to cattle buyers, through local auction barn sales, or through the Virginia Tel-O-Market auction?

The Virginia Beef Cattle Simulation Model (VBCSM) was developed to serve as a decision-aid to assist Virginia cattle producers answer such complex and critical management questions.

3.2. The Virginia Beef Cattle Simulation Model

The Virginia Beef Cattle Simulation Model (VBCSM) is a dynamic, discrete event, stochastic, event-driven,
management simulation model of a user-defined beef cattle operation that retains calves through the stocker growing stage (Figure 3-2). Time incrementation is on a daily fixed-time clock. The model’s dynamic qualities are achieved by using constrained mathematical formulas referred to as algorithms. These algorithms contain the appropriate constants, deterministic variables, and random variables that fall under the control functions for abortions, breeding seasons, calving, cow culling, death loss rates, forage growth and decay, marketing, nutrition, puberty, and replacement breeding females as the beef herd progresses from year -7 to year +3. The model is allowed to approach equilibrium (warm-up) for 8 years and is truncated at the end of year 0. This allows time for the random number generator and stochastic variables to stabilize prior to beginning the first year. At the end of simulation years 1 - 3, the model provides an income statement and a management report for a cow-calf operation in addition to a partial budget and a management report for a stocker calf operation. Since a stochastic model attempts to mimic reality and does not create all animals as equals, variations in VBCSM output occur between simulation years.

Figure 3-3 presents the order in which the various functions from each module are called and executed. “Animal” is the main function that drives VBCSM. The VBCSM includes five primary modules: a forage module, a nutrition
and growth module, a reproduction and life event module, a marketing module, and a tools module. There are 69 functions that utilize 23 stochastic and 798 deterministic variables in 29,809 lines of program code. The animal record structure contains 85 variables.

3.2.1 Forage and Nutrition

Any pasture-based beef cattle production simulation should require that animals interact with the forage production system. In a pasture-based beef cattle production system, performance is dictated by each animal’s body size as well as its other biological characteristics interacting with its immediate environment. The amount of pasture that each animal consumes depends upon the quality, growth and decay of the forage interacting with that animal’s biological traits. However, both the forage and the animal traits must also interact with soil elements such as erosion, mineral fertilization, weed density, slope, terrain, and pasture size. All components must finally interact with management practices in use such as the grazing management system, the stocking density, and the quality and amount of supplemental feed. Animal production can be simulated realistically only when the model accounts for all of these interacting dynamics, each of which can vary over time (Whelan et al., 1984).

In the VBCSM, the forage system is user-defined
according to the region of the state in which the beef cattle operation is located, the erosion potential of the land, the amount of pasture maintenance, the weed infestation of the pasture, the grazing management system in use, the general terrain and slope of the pasture, and forage composition of the pasture. Farm area can be broken into a maximum of six pastures, and the number of acres that are allotted for grazing are allowed to fluctuate monthly depending upon whether or not the pasture is specified as permanent pasture or haylands that are used for grazing between haying seasons. The user is allowed to select from 21 cool and warm season forages common to Virginia to define the forage composition on the farm (Table 3-1). The proportion of the farm composed of any forage species can vary from 0% to 100%. However once the forage composition has been defined, its quality is assumed to represent a typical year. Forage growth fluctuates monthly throughout the year (Table 3-2). Daily senescence modifies the amount of available forage for animal consumption.

Although the herd grazes year around, it can be provided with an optional feed supplement whose composition is user-specified. If the feed supplement option is exercised, the herd will be fed daily supplements as required to make up any nutritional shortfalls. The user can either select from 15 ingredients commonly used in Virginia (Table 3-3) or choose to input up to five
additional ingredients to define the feed supplement fed to the cattle. The proportion of the feed supplement composed of any one ingredient may vary from 0% to 100%. Once the feed supplement has been defined, its quality is assumed to remain constant throughout the simulation.

3.2.2 Reproduction and Life Events

The user specifies whether or not the herd is purebred or crossbred and the number of breeding females. The composition of the breeding herd is defined in terms of twelve beef cattle breeds common to Virginia. Means and standard deviations for several traits in these breeds are given in Table 3-4. Individual animal values for each trait are assigned stochastically by randomly sampling from a normal distribution described by those statistics. The proportion of each breed that is represented in the herd can vary from 0% to 100% depending upon user specifications. The user also specifies the breed or majority breed composition of any replacement females that are to be purchased to maintain herd size and whether or not purebred or crossbred replacements are to be bought.

In order to maximize pre-weaning weight gain in calves, cow-calf producers traditionally breed their cows for spring calving and sell their calves in the fall. Likewise, stocker calves are frequently purchased in the spring and sold in the fall. The VBCSM allows breeding for either
spring or fall calving or both. The day of weaning and the number of days that stockers will be retained are also user-specified. On the day of weaning, the VBCSM sells the weaned calves through the marketing routines to the stocker (retained) calf operation with the exception of any heifers selected to be breeding herd replacements. When the calf has been retained within the stocker herd for the specified number of retention days, it is again sold in the marketing routines and deleted from the herd.

The culling policies practiced on a beef cow-calf operation can influence its profitability. The VBCSM allows the user to choose from four culling policies used in Virginia. These polices are listed in Table 3-5. It has been suggested that the productive efficiency in many beef cattle herds increases as the average age of breeding females decreases and that the critical culling age at which this point is reached is breed dependent (Melton et al., 1994). In the VBCSM, the user can elect to cull breeding females at any age from 8 years to 15 years regardless of a cow’s productive efficiency. However, as culling age decreases, replacement rates necessary to maintain herd size will increase. Also, the incidence of dystocia increases as the culling age is decreased due to the higher proportion of two and three year old females in the breeding herd. Cows are also culled from the breeding herd whenever they reach a body condition score of one.
3.2.3. Marketing

Biological information from VBCSM feeds into a marketing module to simulate financial results for a wide range of management schemes, breeds or breed crosses, forage and nutritional environments. The VBCMS provides Virginia cattlemen with the predicted financial outcome from various management decisions. An enterprise analysis approach is used in which cow-calf production is separated from stocker production and a profit (loss) statement is generated for the cow-calf operation and a partial budget is generated for the stocker calf operation.

The marketing module uses user-specified estimates of the average annual selling prices of 600 - 700 pound feeder steers over a three year period to calculate the market price received for all weights and sex classes of weanling cattle sold. It uses user-specified estimates of the average annual selling prices of 700 - 800 pound yearling cattle for the three year period to calculate the market price received for all weights and sex classes of stocker cattle. It uses the user specified annual selling prices of commercial grade cows for the three year period to calculate the market price received for culled cows. The multipliers used are depicted in Tables 3-6 and 3-7. These multipliers were constructed from historical data published by the United States Department of Agriculture (1996) and data.
provided by the Food and Agricultural Policy Research Institute (FAPRI) in 1997. The average annual price estimates specified are converted into monthly prices by using a monthly price index constructed from the monthly historical prices reported by the Commodity Research Bureau (1997) as a multiplier (Table 3-8).

The VBCSM calculates a sales premium that is based upon breed, color, pregnancy status, and whether the cattle are sold direct to the buyer or through the Virginia Tel-O-Market (Table 3-9). When making direct sales, the VBCSM calculates shrinkage discounts as a pencil allowance, but if selling takes place through either an auction barn or the Virginia Tel-O-Market auction, the shrinkage discount for each animal is calculated as a function of transport distance and overnighting. Additionally, since some Virginia cattle producers use futures market hedging to reduce price risk, the user may elect to hedge either or both the weaned and stocker calves. If hedging is to be used, the user may specify an estimated cash-futures basis between calves that are sold in Virginia and calves that are specified in the Chicago Mercantile Exchange contracts.

Financial results are presented through an income statement for the cow-calf operation and a partial budget for the stocker calf operation. For each type of operation, these reports contain statistics for weaning weight, average daily gain, number of animals of each class sold, death
losses by sex and class, services per conception, conception rates, calving rates, weaning rates, number of pregnant breeding females, and the number of breeding females.

Using an enterprise approach, the cow-calf income statement that is generated by the VBCSM provides a financial summary of what has occurred in the cow-calf enterprise during a one year period. The income statement is made up of a complete set of revenues for each class of livestock (steer calves, heifer calves, orphan calves, and culled cows) that belong to the cow-calf operation, and a set of expenses incurred in their production. Accrued interest is calculated for borrowed capital. Borrowed capital is assumed to represent 80% of the total operational expenses for the cow-calf operation. Only variable expenses are documented in the VBCSM. The income statement may be analyzed to determine the profitability of the cow-calf enterprise given a particular management scheme and its break-even cost of production. After presenting the feed and production summaries of the stocker operation, a sensitivity analysis is presented in an “If-Then” layout that depicts the effects of changes in the price received and the price paid per hundredweight for stocker cattle upon simulated profitability.

The partial budget for the stocker calf operation is more useful than an income statement in analyzing the effects of retaining calves through the stocker stage of
production (Luening et al., 1991). It considers only those changes in income and expense items that are incurred by adding a stocker calf enterprise to the cow-calf operation in order to determine if these changes improve the overall profitability of the cattle operation. The rationale for using a partial budget is that retaining stocker calves has either a positive economic benefit by contributing more revenues than the added cost of retaining them or a negative economic benefit by providing less additional revenues than the added costs that will be incurred by retaining them.

3.2.4. Tools

Stochasticity is achieved in the VBCSM by using random number generators seeded by the computer clock time. The VBCSM use three types of distributions: Bernoulli, uniform, and normal (Steel & Torrie, 1980). In a Bernoulli distribution, the probability of occurrence of a binomial event such as whether a cow does or does not conceive to a mating is assigned according to the value of a randomly sampled variable. In a trial involving Bernoulli variables, there are only two possible outcomes for each trial and the result of each trial is independent of any other trial. A uniform distribution is a distribution in which a random-number generator is called by the program to generate a variate “U” that is distributed continuously and uniformly between zero and one. In VBCSM, uniform values are
calculated and compared to Bernoulli values to determine if an event occurs. For example, in VBCSM, breeding females with parity greater than zero that suffered dystocia during their previous parturition become chronically unable to conceive or maintain a pregnancy longer than 16 days 5.0% of the time (Laster et al., 1973). On the day before the start of the breeding season, a uniform variate is randomly generated for each breeding female suffering dystocia. This variate is compared to 0.05 to determine if the female will be unable to conceive or maintain a pregnancy longer than 16 days. If the randomly generated variable is less than or equal to the Bernoulli value of .05, the female will be unable to conceive. If the number generated is greater than .05, the female is allowed to conceive and maintain a pregnancy longer than 16 days assuming that all other biological conditions for conception are met.

Graphically, a normal distribution is a continuous bell-shaped symmetrical distribution whose center corresponds to the mean and whose height is dependent upon its variance, a small variance giving more height than a large variance (Steel & Torrie, 1980). Many variables in VBCSM are programmed to follow a normal distribution, potential mature body weight for example. An Angus heifer calf could be assigned a potential mature body weight of 1088.56 pounds from a random distribution of potential Angus mature body weights that have a mean mature body weight of
1156 pounds and a standard deviation of 139.9 pounds.

The VBCSM uses the built-in random number generator supplied by the computer to generate values for its uniformly distributed random variables. However, a modification of a portable random number generator more fully described by Press et al. (1988) is used to generate a normal distribution. The portable random number generator used by the VBCSM uses an algorithm provided by Lawrence (1998) to directly convert the normal variate that is generated into the appropriate standard normal value by using the mean and variance from which the stochastic variable is drawn. The portable random number generators use a Box-Muller transformation to convert two uniform deviates generated by the computer’s built-in random number generator that are located in a unit circle into a normal deviate.
LITERATURE CITED


Green, James T. 1998. Personal communication.


Lawrence, J. O. 1998. Personal communication.


National Research Council (NRC). 1996. *Nutrient*


United States Department of Agriculture and Cattle Fax. Cattle industry statistics. USDA. Washington, DC.


Herd Composition:
  - Breeding Cows
  - Replacement Heifers
  - Steer Calves
  - Heifer Calves
  - Culled Cows
  - Stocker Steers
  - Stocker Heifers

Figure 3-1. The General Layout Of The Virginia Beef Cattle Simulation Model.
Figure 3-2. A Diagrammatic Representation Of The Virginia Beef Cattle Simulation Model.

*Arrows Depict The Direction Of Action.
**The Circle Represents The Herd.
***The Large Rectangle Represents The Level Of The Animal.
****The Small Rectangles Represent Functions And Modules.
Figure 3-3. The Virginia Beef Cattle Simulation Model Function Diagram

*Arrows Depict The Direction Of Action And Beginning With ShrinkAnimal, The Order Of Execution.*
<table>
<thead>
<tr>
<th>FORAGE</th>
<th>TDN (%)</th>
<th>METABOLIZABLE ENERGY (MCAL/KG)</th>
<th>NET ENERGY FOR MAINTENANCE (MCAL/KG)</th>
<th>NET ENERGY FOR GAIN (MCAL/KG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FESCUE</td>
<td>58.0</td>
<td>2.100</td>
<td>1.240</td>
<td>0.680</td>
</tr>
<tr>
<td>FESCUE/LADINO CLOVER</td>
<td>56.8</td>
<td>2.128</td>
<td>1.224</td>
<td>0.704</td>
</tr>
<tr>
<td>KENTUCKY BLUEGRASS</td>
<td>66.0</td>
<td>2.400</td>
<td>1.480</td>
<td>0.860</td>
</tr>
<tr>
<td>KENTUCKY BLUEGRASS/WHITE CLOVER</td>
<td>69.0</td>
<td>2.500</td>
<td>1.560</td>
<td>0.940</td>
</tr>
<tr>
<td>ORCHARDGRASS</td>
<td>59.5</td>
<td>2.150</td>
<td>1.290</td>
<td>0.715</td>
</tr>
<tr>
<td>ORCHARDGRASS/LADINO CLOVER</td>
<td>59.7</td>
<td>2.160</td>
<td>1.298</td>
<td>0.725</td>
</tr>
<tr>
<td>ORCHARDGRASS/RED CLOVER</td>
<td>57.7</td>
<td>2.086</td>
<td>1.230</td>
<td>0.661</td>
</tr>
<tr>
<td>ALFALFA</td>
<td>63.6</td>
<td>2.302</td>
<td>1.432</td>
<td>0.847</td>
</tr>
<tr>
<td>HYBRID BERMUDA-100N*</td>
<td>62.5</td>
<td>2.265</td>
<td>1.380</td>
<td>0.740</td>
</tr>
<tr>
<td>HYBRID BERMUDA-200N*</td>
<td>62.5</td>
<td>2.265</td>
<td>1.380</td>
<td>0.740</td>
</tr>
<tr>
<td>BERMUDA/RYE-250N*</td>
<td>63.9</td>
<td>2.319</td>
<td>1.420</td>
<td>0.788</td>
</tr>
<tr>
<td>BERMUDA/RYEGRASS</td>
<td>64.3</td>
<td>2.335</td>
<td>1.432</td>
<td>0.800</td>
</tr>
<tr>
<td>BAHIA GRASS-200N*</td>
<td>53.0</td>
<td>1.920</td>
<td>1.140</td>
<td>0.400</td>
</tr>
<tr>
<td>DALLIS GRASS/LESPEDEZA</td>
<td>64.4</td>
<td>2.330</td>
<td>1.426</td>
<td>0.800</td>
</tr>
<tr>
<td>KUDZA</td>
<td>65.0</td>
<td>2.350</td>
<td>1.440</td>
<td>0.820</td>
</tr>
<tr>
<td>LESPEDEZA</td>
<td>66.5</td>
<td>2.405</td>
<td>1.480</td>
<td>0.860</td>
</tr>
<tr>
<td>RYEGRASS-150N*</td>
<td>67.0</td>
<td>2.440</td>
<td>1.510</td>
<td>0.890</td>
</tr>
<tr>
<td>SERICEA</td>
<td>66.5</td>
<td>2.405</td>
<td>1.480</td>
<td>0.860</td>
</tr>
<tr>
<td>SWITCH GRASS-140N*</td>
<td>52.0</td>
<td>1.890</td>
<td>1.120</td>
<td>0.360</td>
</tr>
<tr>
<td>PEARL MILLETT-150N*</td>
<td>61.0</td>
<td>2.200</td>
<td>1.330</td>
<td>0.680</td>
</tr>
<tr>
<td>SUDAN SORGHUM-150N*</td>
<td>63.0</td>
<td>2.400</td>
<td>1.480</td>
<td>0.860</td>
</tr>
</tbody>
</table>

Table 3-2. Growth Rates In Pounds Per Acre Per Day For Forage Species In The Virginia Beef Cattle Simulation Model (Chamblee & Green, 1995; Green, 1998; Peterson, 1998).

<p>| FORAGE                             | JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC |
|------------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| FESCUE                             | 5   | 10  | 35  | 50  | 38  | 15  | 5   | 8   | 18  | 38  | 13  | 5   |
| FESCUE/LADINO CLOVER               | 4   | 6   | 22  | 54  | 43  | 17  | 9   | 9   | 17  | 22  | 7   | 4   |
| KENTUCKY BLUEGRASS                 | 0   | 1   | 13  | 20  | 33  | 27  | 13  | 5   | 8   | 9   | 2   | 0   |
| KENTUCKY BLUEGRASS/WHITE CLOVER    | 0   | 1   | 13  | 20  | 33  | 27  | 13  | 5   | 8   | 9   | 2   | 0   |
| ORCHARDGRASS                       | 4   | 7   | 28  | 40  | 38  | 16  | 8   | 10  | 14  | 20  | 6   | 4   |
| ORCHARDGRASS/LADINO CLOVER         | 2   | 3   | 24  | 50  | 40  | 20  | 12  | 12  | 14  | 16  | 3   | 2   |
| ORCHARDGRASS/RED CLOVER            | 2   | 4   | 23  | 58  | 47  | 28  | 14  | 14  | 16  | 19  | 4   | 2   |
| ALFALFA                            | 0   | 0   | 27  | 40  | 53  | 40  | 27  | 40  | 27  | 8   | 2   | 0   |
| HYBRID BERMUDA-100N                | 0   | 0   | 0   | 16  | 20  | 30  | 50  | 40  | 30  | 10  | 2   | 0   |
| HYBRID BERMUDA-200N                | 0   | 0   | 0   | 24  | 30  | 45  | 75  | 60  | 45  | 15  | 3   | 0   |
| BERMUDA/RYE-250N                   | 7   | 13  | 37  | 51  | 22  | 29  | 62  | 55  | 33  | 18  | 17  | 11  |
| BERMUDA/RYEGRASS                   | 4   | 6   | 35  | 38  | 54  | 46  | 65  | 58  | 31  | 19  | 14  | 8   |
| BAHIAGRASS-200N                    | 0   | 0   | 0   | 4   | 7   | 39  | 46  | 39  | 29  | 17  | 4   | 0   |
| DALLISGRASS/LESPEDEZA              | 0   | 0   | 0   | 0   | 12  | 30  | 45  | 38  | 23  | 3   | 0   | 0   |
| KUDZA                              | 0   | 0   | 0   | 4   | 18  | 29  | 29  | 18  | 12  | 8   | 0   | 0   |
| LESPEDEZA                          | 0   | 0   | 0   | 4   | 14  | 41  | 35  | 21  | 2   | 0   | 0   | 0   |
| RYEGRAASS-150N                     | 8   | 16  | 30  | 42  | 56  | 47  | 12  | 0   | 0   | 7   | 8   | 7   |
| SERICEA                            | 0   | 0   | 0   | 8   | 23  | 45  | 38  | 23  | 15  | 0   | 0   | 0   |
| SWITCHGRASS-140N                   | 0   | 0   | 5   | 14  | 35  | 47  | 55  | 46  | 24  | 7   | 0   | 0   |
| PEARL MILLET-150N                  | 0   | 0   | 0   | 18  | 50  | 75  | 63  | 38  | 8   | 0   | 0   | 0   |
| SUDAN SORGHUM-150N                 | 0   | 0   | 0   | 19  | 53  | 80  | 67  | 40  | 8   | 0   | 0   | 0   |</p>
<table>
<thead>
<tr>
<th>FEED SUPPLEMENT INGREDIENT</th>
<th>DM (%)</th>
<th>TDN (% DM)</th>
<th>METABOLIZABLE ENERGY (MCAL/KG)</th>
<th>NET ENERGY FOR MAINTENANCE (MCAL/KG)</th>
<th>NET ENERGY FOR GAIN (MCAL/KG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fescue Hay</td>
<td>88.0</td>
<td>58.0</td>
<td>2.10</td>
<td>1.24</td>
<td>0.68</td>
</tr>
<tr>
<td>Fescue/Alfalfa Hay</td>
<td>89.0</td>
<td>56.8</td>
<td>2.13</td>
<td>1.24</td>
<td>0.71</td>
</tr>
<tr>
<td>Fescue/Red Clover Hay</td>
<td>88.4</td>
<td>59.4</td>
<td>2.19</td>
<td>1.30</td>
<td>0.73</td>
</tr>
<tr>
<td>Orchardgrass Hay</td>
<td>89.0</td>
<td>57.0</td>
<td>2.36</td>
<td>1.22</td>
<td>0.66</td>
</tr>
<tr>
<td>Orchardgrass/Alfalfa Hay</td>
<td>89.6</td>
<td>57.8</td>
<td>2.37</td>
<td>1.28</td>
<td>0.71</td>
</tr>
<tr>
<td>Orchardgrass/Red Clover Hay</td>
<td>88.6</td>
<td>57.0</td>
<td>2.25</td>
<td>1.20</td>
<td>0.64</td>
</tr>
<tr>
<td>Lespedeza Hay</td>
<td>90.0</td>
<td>57.0</td>
<td>2.16</td>
<td>1.31</td>
<td>0.73</td>
</tr>
<tr>
<td>Timothy Hay</td>
<td>91.0</td>
<td>58.0</td>
<td>2.06</td>
<td>1.20</td>
<td>0.64</td>
</tr>
<tr>
<td>Alfalfa Hay</td>
<td>90.5</td>
<td>59.0</td>
<td>2.38</td>
<td>1.38</td>
<td>0.79</td>
</tr>
<tr>
<td>Corn (Ground)</td>
<td>87.0</td>
<td>92.0</td>
<td>3.65</td>
<td>2.24</td>
<td>1.55</td>
</tr>
<tr>
<td>Corn Silage (35% Ears)</td>
<td>43.0</td>
<td>72.0</td>
<td>2.92</td>
<td>1.70</td>
<td>1.08</td>
</tr>
<tr>
<td>Soybean Meal (44%)</td>
<td>89.0</td>
<td>85.0</td>
<td>2.90</td>
<td>1.95</td>
<td>1.30</td>
</tr>
<tr>
<td>Broiler Meal</td>
<td>88.0</td>
<td>53.0</td>
<td>1.98</td>
<td>1.33</td>
<td>0.75</td>
</tr>
<tr>
<td>Barley</td>
<td>88.0</td>
<td>85.0</td>
<td>2.91</td>
<td>1.95</td>
<td>1.31</td>
</tr>
<tr>
<td>Oats</td>
<td>89.0</td>
<td>77.0</td>
<td>2.94</td>
<td>1.98</td>
<td>1.33</td>
</tr>
<tr>
<td>Optional, User Defined</td>
<td>XX.X</td>
<td>XX.X</td>
<td>X.XX</td>
<td>X.XX</td>
<td>X.XX</td>
</tr>
<tr>
<td>Optional, User Defined</td>
<td>XX.X</td>
<td>XX.X</td>
<td>X.XX</td>
<td>X.XX</td>
<td>X.XX</td>
</tr>
<tr>
<td>Optional, User Defined</td>
<td>XX.X</td>
<td>XX.X</td>
<td>X.XX</td>
<td>X.XX</td>
<td>X.XX</td>
</tr>
<tr>
<td>Optional, User Defined</td>
<td>XX.X</td>
<td>XX.X</td>
<td>X.XX</td>
<td>X.XX</td>
<td>X.XX</td>
</tr>
</tbody>
</table>
Table 3-4. Descriptive Statistics For Several Traits In The Cattle Breeds Used In The Virginia Beef Cattle Simulation Model (USMARC, 1990).

<table>
<thead>
<tr>
<th>BREED</th>
<th>$\mu$ BIRTH WEIGHT (LBS)</th>
<th>$\sigma$ BIRTH WEIGHT (LBS)</th>
<th>*$\mu$ ADG (LBS)</th>
<th>$\sigma$ ADG (LBS)</th>
<th>$\mu$ MAT. BODY WEIGHT (LBS)</th>
<th>$\sigma$ MAT. BODY WEIGHT (LBS)</th>
<th>$\mu$ GEST. LENGTH (DAYS)</th>
<th>$\sigma$ GEST. LENGTH (DAYS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angus</td>
<td>80.9</td>
<td>9.91</td>
<td>1.87</td>
<td>.16</td>
<td>1156</td>
<td>139.96</td>
<td>285.4</td>
<td>6.02</td>
</tr>
<tr>
<td>Hereford</td>
<td>86.2</td>
<td>10.56</td>
<td>1.70</td>
<td>.145</td>
<td>1188</td>
<td>143.84</td>
<td>288.6</td>
<td>6.09</td>
</tr>
<tr>
<td>Polled Hereford</td>
<td>86.2</td>
<td>10.56</td>
<td>1.70</td>
<td>.145</td>
<td>1188</td>
<td>143.84</td>
<td>288.6</td>
<td>6.09</td>
</tr>
<tr>
<td>Shorthorn</td>
<td>85.8</td>
<td>10.51</td>
<td>1.80</td>
<td>.154</td>
<td>1188</td>
<td>143.84</td>
<td>288.6</td>
<td>6.09</td>
</tr>
<tr>
<td>Charolais</td>
<td>90.6</td>
<td>11.10</td>
<td>1.85</td>
<td>.158</td>
<td>1391</td>
<td>168.42</td>
<td>285.9</td>
<td>6.03</td>
</tr>
<tr>
<td>Chianina</td>
<td>89.3</td>
<td>10.94</td>
<td>1.83</td>
<td>.157</td>
<td>1563</td>
<td>189.24</td>
<td>287.5</td>
<td>6.07</td>
</tr>
<tr>
<td>Gelbvieh</td>
<td>86.0</td>
<td>10.54</td>
<td>1.87</td>
<td>.160</td>
<td>1294</td>
<td>156.67</td>
<td>286.3</td>
<td>6.04</td>
</tr>
<tr>
<td>Limousin</td>
<td>85.8</td>
<td>10.51</td>
<td>1.76</td>
<td>.151</td>
<td>1211</td>
<td>146.62</td>
<td>288.1</td>
<td>6.08</td>
</tr>
<tr>
<td>Maine Anjou</td>
<td>90.6</td>
<td>11.10</td>
<td>1.81</td>
<td>.154</td>
<td>1521</td>
<td>184.16</td>
<td>285.4</td>
<td>6.02</td>
</tr>
<tr>
<td>Pinzgauer</td>
<td>86.4</td>
<td>10.58</td>
<td>1.76</td>
<td>.151</td>
<td>1232</td>
<td>149.17</td>
<td>286.0</td>
<td>6.03</td>
</tr>
<tr>
<td>Simmental</td>
<td>88.9</td>
<td>10.89</td>
<td>1.83</td>
<td>.157</td>
<td>1296</td>
<td>156.91</td>
<td>286.3</td>
<td>6.04</td>
</tr>
<tr>
<td>Tarentaise</td>
<td>82.7</td>
<td>10.13</td>
<td>1.81</td>
<td>.154</td>
<td>1134</td>
<td>137.30</td>
<td>287.1</td>
<td>6.06</td>
</tr>
</tbody>
</table>

*Indicates ADG For Pasture-Based Effects
Table 3-5. Alternative Culling Polices Simulated In The Virginia Beef Cattle Simulation Model.

<table>
<thead>
<tr>
<th>VBCSM CULLING POLICIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open when pregnancy tested</td>
</tr>
<tr>
<td>Fail to calve or wean a calf in any one year</td>
</tr>
<tr>
<td>Fail to calve or wean a calf in any two years</td>
</tr>
<tr>
<td>Fail to calve or wean a calf in any three years</td>
</tr>
</tbody>
</table>
Table 3-6. Multiplicative Price Adjustments Used To Calculate The Price Of Different Sexes And Weight Classes Of Weaned Cattle In The Virginia Beef Cattle Simulation Model. (FAPRI, 1997; Commodity Research Bureau, 1997)

<table>
<thead>
<tr>
<th>Livestock Class</th>
<th>Multiplicative Price Adjustments</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 OR LESS LB. HEIFERS</td>
<td>0.89772</td>
</tr>
<tr>
<td>400 – 500 LB. HEIFERS</td>
<td>0.90813</td>
</tr>
<tr>
<td>500 – 600 LB. HEIFERS</td>
<td>0.90537</td>
</tr>
<tr>
<td>600 – 700 LB. HEIFERS</td>
<td>0.88218</td>
</tr>
<tr>
<td>700 – 800 LB. HEIFERS</td>
<td>0.86462</td>
</tr>
<tr>
<td>800 – 900 LB. HEIFERS</td>
<td>0.83520</td>
</tr>
<tr>
<td>900 – 1000+ LB. HEIFERS</td>
<td>0.79980</td>
</tr>
<tr>
<td>400 OR LESS LB. STEERS</td>
<td>1.10700</td>
</tr>
<tr>
<td>400 – 500 LB. STEERS</td>
<td>1.06400</td>
</tr>
<tr>
<td>500 – 600 LB. STEERS</td>
<td>1.05200</td>
</tr>
<tr>
<td>600 – 700 LB. STEERS</td>
<td>1.00000</td>
</tr>
<tr>
<td>700 – 800 LB. STEERS</td>
<td>0.97189</td>
</tr>
<tr>
<td>800 – 900 LB. STEERS</td>
<td>0.93980</td>
</tr>
<tr>
<td>900 – 1000+ LB. STEERS</td>
<td>0.90000</td>
</tr>
</tbody>
</table>
Table 3-7. Multiplicative Price Adjustments Used To Calculate The Price Of Different Sexes And Weight Classes Of Stocker Cattle In The Virginia Beef Cattle Simulation Model. (FAPRI, 1997; Commodity Research Bureau, 1997).

<table>
<thead>
<tr>
<th>Livestock Class</th>
<th>Multiplicative Price Adjustments</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 OR LESS LB. HEIFERS</td>
<td>0.9237</td>
</tr>
<tr>
<td>400 – 500 LB. HEIFERS</td>
<td>0.9344</td>
</tr>
<tr>
<td>500 – 600 LB. HEIFERS</td>
<td>0.9316</td>
</tr>
<tr>
<td>600 – 700 LB. HEIFERS</td>
<td>0.9077</td>
</tr>
<tr>
<td>700 – 800 LB. HEIFERS</td>
<td>0.8896</td>
</tr>
<tr>
<td>800 – 900 LB. HEIFERS</td>
<td>0.8594</td>
</tr>
<tr>
<td>900 – 1000+ LB. HEIFERS</td>
<td>0.8229</td>
</tr>
<tr>
<td>400 OR LESS LB. STEERS</td>
<td>1.1390</td>
</tr>
<tr>
<td>400 – 500 LB. STEERS</td>
<td>1.0950</td>
</tr>
<tr>
<td>500 – 600 LB. STEERS</td>
<td>1.0820</td>
</tr>
<tr>
<td>600 – 700 LB. STEERS</td>
<td>1.0290</td>
</tr>
<tr>
<td>700 – 800 LB. STEERS</td>
<td>1.0000</td>
</tr>
<tr>
<td>800 – 900 LB. STEERS</td>
<td>0.9670</td>
</tr>
<tr>
<td>900 – 1000+ LB. STEERS</td>
<td>0.9260</td>
</tr>
</tbody>
</table>
Table 3-8. The Monthly Pricing Index Used To Calculate The Price Of Cattle In The Virginia Beef Cattle Simulation Model (Commodity Research Bureau, 1997).

<table>
<thead>
<tr>
<th>MONTH</th>
<th>MULTIPLICATIVE MONTHLY ADJUSTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>1.015</td>
</tr>
<tr>
<td>February</td>
<td>1.045</td>
</tr>
<tr>
<td>March</td>
<td>1.054</td>
</tr>
<tr>
<td>April</td>
<td>1.041</td>
</tr>
<tr>
<td>May</td>
<td>1.019</td>
</tr>
<tr>
<td>June</td>
<td>1.011</td>
</tr>
<tr>
<td>July</td>
<td>0.997</td>
</tr>
<tr>
<td>August</td>
<td>1.009</td>
</tr>
<tr>
<td>September</td>
<td>0.968</td>
</tr>
<tr>
<td>October</td>
<td>0.948</td>
</tr>
<tr>
<td>November</td>
<td>0.942</td>
</tr>
<tr>
<td>December</td>
<td>0.950</td>
</tr>
</tbody>
</table>
Table 3-9. Multiplicative Price Adjustments Used To Calculate The Selling Price Of Cattle In The Virginia Beef Cattle Simulation Model (FAPRI, 1997; Commodity Research Bureau, 1997).

<table>
<thead>
<tr>
<th>TYPE OF PREMIUM</th>
<th>MULTIPLICATIVE PRICE ADJUSTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breed Premium, Angus Heifer</td>
<td>1.0042719</td>
</tr>
<tr>
<td>Breed Premium, Angus Steer</td>
<td>1.0215270</td>
</tr>
<tr>
<td>Black (not Angus) Heifer Premium</td>
<td>1.014845</td>
</tr>
<tr>
<td>Black (not Angus) Breed Steer Premium</td>
<td>1.016717</td>
</tr>
<tr>
<td>Breed Premium, Angus Culled Cow</td>
<td>1.00565</td>
</tr>
<tr>
<td>Black (not Angus) Culled Cow Premium</td>
<td>1.0136</td>
</tr>
<tr>
<td>Pregnant Replacement Heifers</td>
<td>1.1200</td>
</tr>
<tr>
<td>Direct Sales</td>
<td>1.0500</td>
</tr>
<tr>
<td>Virginia Tel-O-Market Sales</td>
<td>1.0650</td>
</tr>
</tbody>
</table>
Chapter 4 Reproduction and Life Events

Module

4.1. Introduction
4.2. UseInput1
4.3. UseInput2
4.4. UseInput11
4.5. UseInput2A
4.6. UseInput2B
4.7. CreateHerd
4.8. GetHeadAnimal
4.9. HerdHead
4.10. ShrinkAnimal
4.11. GetAnimal
4.12. Dystocia
4.13. Sterility
4.14. Parturition
4.15. CalfMilk
4.16. Puberty
4.17. PregnancyTest
4.18. CullCows
4.19. Lactation
4.20. BreedFemales
4.22. AddAnimal
4.23. LastID
4.24. Abort
4.25. Mortality
4.26. SBBreeders
4.27. GetSex
4.28. Wean
4.29. ChangeSex
4.30. ReproStats
4.31. DeadCattle
4.32. DeleteAnimal
4.33. DumpList
4.34. ResetCowVariables
4.35. UseYearOption
4.36. DeleteHerd
4.37. ResetCowConCntr

Literature Cited
Chapter 4

Reproduction and Life Events Module

Abstract: The reproduction and life events module (Cow) in the Virginia Beef Cattle Simulation Model (VBCSM) contains functions that simulate the conception of the fetus, the fertility of the cows, the culling policy practiced on the breeding herd, the type and source of replacement heifers, dystocia, "sterility" (failure to conceive), parturition, puberty, abortions, mortality, weaning, and transfer of calves to the stocker herd. Many of the variables used in this module are stochastic. Except for user supplied values, the means and standard deviations for such variables were derived from published research studies.

4.1. Introduction

The purpose of this chapter is to describe the reproduction and life events module (Cow) of the Virginia Beef Cattle Simulation Model (VBCSM). This will be accomplished by discussing each major Cow function that impacts the VBCSM model and the concepts that justify them in the order in which they are executed by VBCSM. Appropriate tables will accompany the functions discussed.

4.2. UseInput1

UseInput1 stores information gathered in the first page of dialog (Figure A-3, APPENDIX A) under the "Input Herd Values" prompt of the "Main Menu" (Figure A-2, APPENDIX A) prompt. It stores the name of the farm, its address, the name of its owner, the name of its farm manager, and the
name of its herdsman or livestock manager.

4.3. UseInput2

UseInput2 stores information furnished by the user in the second page (Figure A-4, APPENDIX A) of dialog under the "Input Herd Values" prompt in the "Main Menu." Among the information obtained is whether or not the herd is artificially inseminated, if ionophores are provided, and if calves are implanted. Other information obtained determines if purebred or crossbred and bred or open heifers are purchased as replacements, their breed, and if replacements are first selected within the herd before any purchases are made. UseInput2 also stores the size of the breeding herd, the number, cost and salvage value of its bulls as well as how long each bull is kept. It additionally stores the culling policies to be used during the simulation, and if the herd is mated to produce black calves.

4.4. UseInput11

From the third page of dialog (Figure A-5, APPENDIX A) under the "Input Herd Values" prompt in the "Main Menu", information obtained through UseInput11 determines the number of days that stocker calves will be retained and whether the simulation program calculates the post-weaning average daily gain of the stocker cattle or whether it uses a user-provided annual or seasonal post-weaning average daily gain.
4.5. **UseInput2A**

From the fourth page of dialog (Figure A-6, APPENDIX A) under the “Input Herd Values” prompt in the “Main Menu”, information obtained through UseInput2A determines if the breeding herd is purebred or crossbred and its breed if it is purebred or its breed composition and mating program if it is not. If the herd is designated as crossbred, the user is allowed to select one breed of bulls to mate to the more mature females of the herd and a bull from another breed to mate to the herd’s younger females.

4.6. **UseInput2B**

From the fifth page of dialog (Figure A-7, APPENDIX A) under the “Input Herd Values” prompt in the “Main Menu”, information obtained through UseInput2B determines whether or not cattle will be bred to calve during the spring, the fall, or to calve in both spring and fall. It also stores the length of each breeding season, the first day of each breeding season, the first day of each heifer breeding season, and the annual weaning day for spring-born calves and fall-born calves.

4.7. **CreateHerd**

CreateHerd is the first event executed in VBCSM. This function creates the initial breeding herd. These cows may be either purebred or crossbred as selected by the user from 12 common breeds found in Virginia. Purebred herds are
created from breed-appropriate distributions, while the
typical cow from a crossbred herd can be composed of several
different breeds. Each cow created is assigned an age drawn
from a normal distribution with a mean of 1730 days (4.78
years) according to Azzam et al. (1990a) and a standard
deviation of 275 days. If a cow’s age is less than 731
days, she is assumed to be non-parous. If a cow’s age is
between 731 days and 1096 days, she is assigned a parity of
one. If a cow’s age is 1096 days or older, she is assigned
a parity of two. Each cow is assigned a Body Condition
Score (BCS) of five. Each is assumed to be nonpregnant and
non-lactating. Production variables such as birth weight
(BWT), potential average daily gain (PADG), potential mature
body weight (PMWT), and gestation length (PGEST) are
stochastic in nature and are assigned from breed appropriate
normal distributions for a purebred (see Table 3-4). Each
purebred cow is additionally assigned a potential slaughter
weight (PMSWT) also drawn from a breed appropriate normal
distribution. The potential slaughter weight value is used
to calculate the equivalent shrunk body weight (EQSBW) of
non-replacement calves. Table 4-1 displays the mean values
and standard deviations for the potential slaughter weight
of each breed. The current weight of each cow is allowed to
fluctuate daily. This function also assigns a sex
designation of three to each cow and an identification
number that allows the animal to be tracked. The breed of
the herd sire must equal the breed of the herd cows for a purebred herd.

For a crossbred herd, the values of BWT, PADG, PMWT, PGEST, AND PMSWT for the cows are drawn from a normal distribution with a mean and a standard deviation that are calculated using the typical cow’s user-specified breed composition. The breed of the herd sire can be selected from any of the 12 common breeds found in Virginia.

4.8. **GetHeadAnimal**

This is a standard C++ function used to retrieve the record of the animal that is at the head of the linked list whenever the linked list must be counted or searched.

4.9. **HerdHead**

HerdHead is another standard C++ function that retrieves the record of the head animal on the linked list whenever it is necessary to count through the entire linked list.

4.10. **ShrinkAnimal**

The VBCSM calculates each animal’s daily weight in both pounds for use in the marketing functions and kilograms for use in the nutrition functions. These variables are also calculated daily in the nutrition function, DWGAIN. The algorithms used to calculate the weights needed by the simulation are represented as,

\[
SBWT = CDWT * 0.96
\]
PSBWMK_{Steers} = PMSWTK * 0.96
PSBWMK_{Heifers} = PMWTK * 0.96
EBWTK = SBWTK * .891
EBWTK_{Pregnant Cows} = (SBWTK - GUWTK) * .891
EQSBW = SBWTK * (SRW/PSBWMK)
EQSBW_{Pregnant Cows} = (SBWTK - GUWTK) * (478/PSBWMK)
EQEBW = EQSBW * .891

where,
CDWTK = The current day’s live body weight, Kg.
EBWTK = The current day’s empty live body weight, Kg.
EQSBW = The NRC equivalent shrunk body weight of an animal, Kg. It is used to predict average daily gain in non-replacement calves.
EQEBW = The NRC equivalent empty body weight of an animal, Kg. It is used to predict average daily gain in non-replacement calves.
GUWTK = The weight of the gravid uterus, Kg.
PSBWMK = The potential shrunk slaughter weight of a non-replacement animal, Kg.
PMSWTK = The potential slaughter weight of
a non-replacement animal, Kg.

PMWTK = The potential mature weight of a breeding herd female, Kg.

SBWTK = The current day’s shrunk body weight, Kg.

SRW = A standard reference weight for an animal finishing at small marbling (28% body fat), Kg.

4.11. GetAnimal

GetAnimal is used by other functions to retrieve the record of a particular animal by its identification number. It is often used in mother-calf routines.

4.12. Dystocia

Dystocia or difficult parturition is an economic problem in the beef cattle industry because it is a primary cause of calf mortality (Bellows et al., 1971; Laster et al., 1973; Laster, 1974). Dystocia also influences conception in the subsequent reproductive cycle (Laster et al., 1973; Laster, 1974).

Dam age and calf birth weight are the two most important factors that influence dystocia (Laster et al., 1973). Calf birth weight is influenced by genotype, calf sex, length of gestation, and dam age (Laster, 1974). Laster (1974) reported that although pelvic size independent of cow weight had a significant influence on dystocia, it
was not a large source of accountable variation. Studies also indicate that a large portion of the differences in pelvic size, particularly among breeds, probably result from differences in body size. Bellows et al. (1971) reported that the pre-calving body weight of the dam exerted an effect on birth weight. Studies have also shown that male calves have longer gestation periods and higher birth weights than female calves. This may account for the increased incidence of dystocia in cows giving birth to male calves compared to female calves (Bellows et al., 1971). Rice (1994) reported that while inadequate nutrition can lower birth weights, dystocia rates are not altered even for heifers unless they are small, weak, or excessively fat. Perhaps, dystocia could be more accurately defined as any birth that reduces calf viability, causes maternal injury, or requires assistance.

The VBCSM models dystocia in the breeding herd as occurring whenever:

\[ \text{MCDYSF} \leq \text{DYSTF} \]

where MCDYSF is a value drawn from a uniform distribution, and DYSTF is equal to a base probability that dystocia will occur, plus an adjustment of one percent more or one percent less for each pound of birth weight above or below an established birth weight for the age and genetic composition of the dam including heterotic effects, if any, and the sex of its calf. A mathematical description of DYSTF using the
appropriate variables found in Table 4-2 is:

\[ \text{DYSTF} = \text{DYPer}(M/F)X + (0.01 \times (\text{Calf Birth Weight} - \text{ACW}(M/F)X) \]

\[ M/F = \text{Sex of Calf} \quad X = \text{Age of Dam} \]

This algorithm incorporates the findings of Laster et al. (1973) in that the probability of calving difficulty varies by an approximate average of 1.0% for each additional pound of birth weight above or below the established average birth weight for the calves of a particular sex from dams of a particular age. This algorithm, however, has been adjusted to reflect the characteristics of a 1999 cow as opposed to a 1973 cow.

4.13. Sterility

In VCBSM, sterility is defined as the chronic inability to conceive or maintain a pregnancy longer than 16 days. In its reproductive model, Cornell University as reported by Azzam et al. (1990b) assumed that 10% of all cows with abnormal parturitions afterwards became unable to conceive or maintain a pregnancy. Johnson & Notter (1987) in their reproductive model assumed that 5% of the cows would be infertile and that the infertility resulted from environmental effects rather than additive genetic effects.

In modeling reproductive systems, Azzam et al. (1990b) assumed three percent of all heifers were sterile. Saacke (1996) and Beal (1997) indicated that perhaps as many as
five percent of non-parous heifers may be unable to conceive or maintain a pregnancy. In 1958, Lindley et al. reported that after 8.8 services, 3.2% of their cows failed to conceive, while Milagres et al. (1979) found that after two consecutive years, 19% of their yearling heifers still had not conceived.

The VBSCM incorporates the findings of Lindley et al. (1958) by assuming that 3.2 percent of the non-parous heifers simulated are sterile. For simulated cows having a parity of one or more, the VBCSM assumes that 5.0% of cows may become permanently barren after experiencing dystocia as a result of pathological conditions such as metritis and structural damage to the reproductive tract.

4.14. Parturition

The parturition function allows each pregnant female to give birth to a live calf at the end of the gestation period. The pregnant cow gives birth to its calf whenever the number of days that it has been pregnant becomes equal to or greater than its stochastically assigned length of gestation. The act of giving birth depends upon two animal structure variables, DPREG and PGEST. DPREG is a counter that tracks the number of days that the animal has been pregnant, i. e., the number of days since the animal conceived. PGEST is a randomly generated number from a breed appropriate normal distribution (see Table 3-4) for a
purebred cow that represents the termination of that cow’s current gestation. If the cow is crossbred, PGEST is a randomly generated number from a normal distribution that is appropriate for the cow’s genetic composition. Mathematically, birth occurs when:

\[
\text{DPREG} \geq \text{PGEST}
\]

After parturition, the cow is allowed to start cycling again. However, a postpartum interval or period of time that allows the cow to recuperate after parturition must pass before her first fertile estrus can occur. It’s length may be influenced by the breed of the cow or its genetic composition, its parity, whether or not the cow suffered dystocia, lactation effects, and nutrition (Short et al., 1994). In properly managed beef cattle, postpartum intervals normally range from 35 days to 70 days. In nutritionally stressed cows, they may exceed 100 days. In fact, the length of the postpartum interval is most affected by nutrition and suckling (Short et al., 1990). Azzam et al. (1990b) modeled the postpartum interval as a normal distribution with a mean of 65 days and a standard deviation of 18 days, while Johnson and Notter (1987) reported that phenotypes for postpartum interval were not normally distributed. Short et al. (1972) indicated that cattle may not be physiologically able to reinitiate estrous cycles before 25 to 30 days postpartum. Relationships among the physiological events that occur during the postpartum
interval are quite complex. Perhaps such traits as body weight and body condition are important only because they are related to the physiological events (Greer et al., 1990). Greer et al. (1990) also indicated the first fertile estrus in cows that did conceive occurred at an average of 54.4 days postpartum.

The VBCSM assumes that the breeding herd is not nutritionally stressed and models the postpartum interval as a symmetrical distribution with a mean of 54.4 days and a standard deviation of 12.5 days. It will continue to recalculate a postpartum interval for a cow until the random value generated is equal to or greater than 30 days. If the cow is less than three years old at the time of parturition, seven days will be added to its postpartum interval. If the cow experienced dystocia, another 14 days will be added to its postpartum interval.

At parturition, pregnancy counters are reset to zero while a counter that represents the number of days that the cow has lactated is activated. The animal’s current shrunk body weight in kilograms is recalculated by subtracting the weight of the contents of the gravid uterus. All other weights are then recalculated. The variables in the cow’s record structure that relate to the fetal identification number, its calf’s birth weight, the gravid uterus weight, and the sex of its calf are reset to zero while the variable representing its calf’s identification number is set to
equal the fetal identification number. The calf’s record is then retrieved and activated by advancing its age from zero to one. A counter named TotGestLen used to calculate the average length of gestation in the herd in the function ReproStats is also advanced by adding the PGEST of each cow at parturition.

4.15. CalfMilk

CalfMilk transfers the value of a cow’s daily milk yield (YN) in kilograms (Kg/day) and the total daily energy (YEN) provided by that milk in mega-calories per kilogram (Mcal/kg) to its calf. Those values will be used in the nutrition function, DWGAIN.

4.16. Puberty

Puberty calculates the date of the pubertal estrus for each heifer that is designated as a breeding herd replacement. Puberty is the time at which sexual maturity is attained. Heifer weight is important because a threshold weight must be attained before the pubertal estrus occurs. Body weight and age are highly correlated in maturing animals. Whenever a restricted breeding season is used, the age at which heifers reach puberty becomes important if they are to be bred to calve as two year olds (Ferrell, 1982). Normally, dual purpose beef cattle breeds that are selected for both high levels of milk production and beef production arrive at puberty at a younger age and a lighter weight.
proportional to their mature weight than do breeds selected only for beef production (Ferrell, 1982; Koch et al., 1994). Some of these differences may be related to maternal effects exerted through higher pre-weaning gains in calves from the higher milk producing breeds (Ferrell, 1982). Koch et al. (1994) reported that pubertal weights of English beef cattle breeds were lighter relative to their mature weights than Continental non-dual-purpose beef breeds. Bolze and Corah (1993) reported that heifers on average attain 65% of their mature body weight before showing signs of estrus. The National Research Council (1996), however, reported that heifers of typical beef breeds usually express puberty when they reach 60% of their mature body weight while those from dual purpose breeds are inclined to express puberty at approximately 55% of their mature weight.

VBCSM uses the NRC (1996) data by modeling the date of the pubertal estrus as occurring when heifers of Angus, Hereford, Polled Hereford, Shorthorn, Charolais, Chianina, Limousin, Maine-Anjou, and Tarentaise reach 60% of their potential mature weight. Pubertal estrus occurs in heifers of Gelbvieh, Pinzgauer, and Simmental breeding when they attain 55% of their potential mature body weight. VBCSM calculates the date of pubertal estrus only for those heifers selected to be breeding replacements.
4.17. **PregnancyTest**

The VBCSM allows cows to be culled for a variety of reasons, including culling nonpregnant cows if the herd is pregnancy tested (Table 3-5). The PregnancyTest routine is also used to implement the CullCows function explained in section 4.18.

It is a good practice to pregnancy test the breeding herd after the end of the breeding season in order to cull non-pregnant cows. By culling unproductive cows, the costs of maintaining them until the next breeding season are eliminated while creating the opportunity to improve the genetic merit and the future economic efficiency of the herd (Warnick, 1994). Most pregnancy testing is done 60 days after the end of breeding season (Bellows and Short, 1994).

In VBCSM, PregnancyTest is the Cow function that examines each cow, at least 56 weeks old, 60 days after the end of each breeding season for pregnancy. If the selected culling policy is to cull nonpregnant cows at the time that pregnancy testing is conducted, a counter in each nonpregnant animal’s record structure called SEAOP is advanced. SEAOP tracks the number of breeding seasons that a cow has been open. A second counter called TSTOP is also advanced. TSTOP counts the number of nonpregnant cows in the herd that have not yet been replaced.

If another culling policy is used, VBCSM continues to
pregnancy test the herd, but the animal record structure named FAIL is advanced rather than the SEAOP variable. FAIL is a record structure counter that tracks the number of years that a female fails to calve. On the last day of the simulation year, a general counter named CowsFail that indicates the number of cows that failed to calve during the year is advanced. Depending upon the culling policy selected, females that failed to calve and who meet the culling criterion are removed from the breeding herd on the day before the first day of the next simulation year’s breeding season.

If more than one annual breeding season is used to implement culling decisions and the herd is to be pregnancy tested, then when a cow is pregnancy tested and found to be open after the end of the first breeding season, the animal record structure variable called SEAOP1 is advanced to one. After the end of the second breeding season in which the cow is found to be open, another animal record structure variable called SEAOP2 is also advanced to one. Whenever SEAOP1 plus SEAOP2 are equal to two, the animal record structure variable FAIL is advanced and the earliest variable, SEAOP1 or SEAOP2, is reset to zero. However, if the animal becomes pregnant, both variables are reset to zero. The same process applies if the user elects not to pregnancy test the herd, but the cow is allowed to remain in the herd until the user-specified culling criterion is met.
4.18. **CullCows**

Culling is the removal of one or more animals from the breeding herd. Cow culling and replacement strategies can strongly influence herd efficiency and profit (Melton, 1980). The key to success is rigid culling (Juergensson, 1980). Cows that fail to breed should be culled. Pregnancy testing may be used to improve the reproductive efficiency of the herd by identifying the cows that failed to become pregnant. By culling cows for reproductive performance, a selective pressure is maintained that results in the early removal of females that are marginally fertile, unthrifty, or poorly adapted to the local environment. It has also been reported that if profit maximization is used as the standard criterion for economic analysis, the optimum culling age differs for different breeds (Melton et al., 1993). Azzam and Azzam (1991) reported that in fall calving systems, average net revenues in the long run are maximized when all open cows are culled and replaced by pregnant heifers although not all studies support this conclusion.

CullCows is the Cow function that culls cows in compliance with a user-specified culling policy selected from four culling options (see Table 3-5) and with a user-specified culling age. If the culling policy is anything other than culling at the time of pregnancy testing and the breeding season being simulated is for the spring calving
season, no cows will be culled before the second simulation year of the warm-up period, Year -6. If the breeding season for fall calving is selected, no cow is culled before the third simulation year of the warm-up period, Year -5. Otherwise, cows are marked for culling on December 31 if the total of their record structure variables FAIL plus FWEAN equals 1, 2, or 3 for culling policies 1, 2, or 3, respectively.

Whenever a cow fits the culling criteria or whenever its body condition score has dropped to a one, its record structure variable named CULL is advanced to a one. Two general counters are also advanced. One is called REPLR and it indicates the number of replacement heifers that are required. The other is called CulledCows and it indicates the number of culled cows that currently remain in the breeding herd. Whenever a pregnant cow is marked for culling, the animal record of its fetus is deleted from the linked list. Every cow marked for culling remains in the herd until it is no longer lactating. When the lactation period is over, a counter named CLNUM that represents the number of cows culled during the current year is advanced and the animal is sold.

4.19. Lactation

Lactation ensures that every cow that has nurtured a calf is no longer lactating on the day after the calf is
weaned. It is used whenever the current simulation date is
greater than the Julian calendar day on which weaning takes
place, and the cow still has a calf identification number
greater than zero. If this event occurs, the variables
relating to the cow’s lactation status, its calf’s
identification number, the number of its consecutive
lactating days, and its variables for milk fat, milk solids,
milk yield, and the total energy provided by its milk are
reset to zero.

4.20. BreedFemales

Economic efficiency and profitability in a cow-calf
operation depend upon the reproductive efficiency of the
breeding herd. Wiltbank (1984) stated “Reproduction in a beef cow herd is a fragile thing, easy to disrupt, and
difficult to re-establish.” Five factors determine whether
or not a parous female will display estrus during the first
20 days of the breeding season. These are the time of
calving, the body condition score of the cow, the age of the
cow, whether or not it suffered dystocia in the previous
pregnancy, and whether or not it is nursing a calf. Body
condition score has a significant effect upon a cow becoming
pregnant. Rae et al. (1993) reported that parity and body
condition score both significantly influenced pregnancy
rates in beef cows located in Florida. Kunkle et al. (1994)
reported that a critical body condition score of five was
necessary to achieve high pregnancy rates. When cows had a body condition score less than five, they conceived 32% less often than cows with a score of five. Those with a body condition score greater than five became pregnant 12% more often.

Frequently, poor reproductive rates result in long calving seasons (Wiltbank, 1983). The only way to be certain that cows have sufficient time to show estrus is to have a short calving season. Since late calving heifers become late calving cows, they should be bred earlier than the regular herd. In fact, Wiltbank (1983) helped to devise a system in which the breeding season for heifers started 20 days earlier than that for the regular herd.

Estrus will recur in most cows every 18 to 24 days. First service conception rates reported in beef cattle vary widely. Wiltbank et al. (1961) reported conception rates during the first 21 days of the breeding season at the Beef Cattle Research Station at Front Royal, Virginia to range from 65% to 84% depending upon age and breed of the cow. However, this study failed to measure the number of cows that conceived but aborted the fetus prior to the 16th day of pregnancy. Azzam et al. (1990b) used first service conception rates as high as 80% in modeling a reproductive management system for beef cattle.

BreedFemales is the function used to mate the cows during the breeding season and to determine if they
conceive. If the cow has a body condition score (BCS) of five, it has a 75% baseline chance of conceiving at each estrus (CONCV), modified by an adjustment that accounts for the probability that she will abort during the first 16 days of pregnancy (to be explained in Section 4.24.). This “abortion effect” is multiplicative and modeled as,

\[ 1 + \{0.6111 \times (0.115 + (0.01 \times \text{dam’s age in years})\) \}\]

For example, using this algorithm, the probability that a 10 year old cow with a body condition score of five would conceive is 84.85% and is calculated as,

\[ 75\% \times [1 + \{0.6111 \times (0.115 + (0.01 \times 10))\}] = 84.85\% \]

If the female’s body condition score (BCS) is less than a five, its base line chance to conceive during each estrus period (CONCV) is reduced multiplicatively by 31.76% to 51.18% following the indications of Kunkle et al. (1994). If its body condition score (BCS) is greater than five, its base line chances to conceive (CONCV) multiplicatively increase by 11.77% to 83.83%. If the cow suffered dystocia on her last parturition, the probability of conception is reduced additively by 6.1% following the indications of Laster et al. (1973).

When simulating conception, the length of the cumulative estrus cycles must never exceed 365 days at the end of each year but the simulation requires continuity to be maintained. Therefore on the first day of each simulation year, each cycling female has the variable
representing the cumulative length of its estrous cycles (CUCYC) reset. This can occur in different ways under different conditions. Symbolically, this is represented under the different conditions as follows,

For a female that reached puberty during the preceding year,

\[
\text{If } (\text{PUBER} > 1) \text{ and if } (\text{PUBER} + \text{CUCYC}) \geq 365)\)
\]

\[
\text{CUCYC} = \text{CUCYC} + \text{PUBER} - 365
\]

\[
\text{CNDAT} = 0
\]

\[
\text{PUBER} = 1
\]

For a female reaching puberty in a year previous to the preceding year,

\[
\text{IF } (\text{PUBER} = 1) \text{ and IF } (\text{CADAT} + \text{CUCYC}) \geq 365)\)
\]

\[
\text{CUCYC} = \text{CUCYC} + \text{CADAT} - 365
\]

\[
\text{CNDAT} = 0
\]

\[
\text{PUBER} = 1
\]

where,

\[
\text{CADAT} \quad = \quad \text{The female’s latest (immediately previous) calving date. This has a value of zero if the female did not give birth in the previous year.}
\]

\[
\text{CUCYC} \quad = \quad \text{The cumulative length of the female’s estrous cycles.}
\]

\[
\text{PUBER} \quad = \quad \text{the date of the female’s pubertal estrus in Julian calendar days. This is equal to a one if}
\]
the female’s pubertal estrus occurred before the previous year.

\[ CNDAT = \text{The conception date for the female’s latest (immediately previous) gestation period. This has a value of zero if the female was not pregnant in the previous year.} \]

After a female has reached puberty, then whenever the current simulation day \((\text{currentDATE})\) is equal to or greater than the sum of one plus the length of its cumulative estrus cycles \((\text{CUCYC})\) plus its latest calving date \((\text{CADAT})\), another estrous cycle \((\text{MCCYC})\) is drawn from a normal distribution and added to the length of the cumulative estrus cycles \((\text{CUCYC})\). This continues as long as she is neither pregnant nor “sterile.”

Fertile estrus \((\text{FESTR})\) begins to occur in a non-parous, cycling cow that reached puberty during the current year whenever \(\text{currentDATE}\) is equal to or greater than the date of the cow’s pubertal estrus \((\text{PUBER})\). If, however, the pubertal estrus \((\text{PUBER})\) occurred previous to the current simulation year, the fertile estrus \((\text{FESTR})\) is equal to \(\text{CUCYC}\). Fertile estrus \((\text{FESTR})\) is represented in non-parous females that reached puberty during the current year as,

\[
\text{If } (\text{PUBER} > 1) \\
\text{FESTR} = \text{PUBER} + \text{CUCYC}
\]
Fertile estrus (FESTR) is represented in non-parous females the reached puberty in a prior year as,

\[ \text{If (PUBER = 1)} \]

\[ \text{FESTR} = \text{CUCYC} \]

In a parous female, a fertile estrus (FESTR) occurs whenever the current simulation day (currentDATE) becomes greater than the cow’s last calving day plus the length of its cumulative estrous cycles (CUCYC). Symbolically, this is represented as,

\[ \text{FESTR} = \text{CADAT} + \text{CUCYC} \]

For females having parity equal to zero that are eligible to breed, breeding begins as soon as the current simulation day (currentDATE) is equal to or greater than the first day of the breeding season for heifers (FDBHSX). For cows with a parity greater than zero, breeding begins as soon as the current simulation day (currentDATE) is equal to or greater than the first day of the regular breeding season (FDBSX). Breeding for heifers continues until the last day of the heifer breeding season which is calculated as the first day of the heifer breeding season (FDBHSX) plus the length of the breeding season (SEASX). Cows are bred until the last day of the regular breeding season (LDBSX) which is calculated as the first day of the regular breeding season (FDBSX) plus the length of the breeding season (SEASX). By ending the breeding season for heifers earlier than the breeding season for more mature cows, the heifer is allowed
additional time to complete her postpartum interval prior to entering the next breeding season.

During the breeding season and upon reaching puberty, if the heifer is at least 56 weeks of age (13 months), she is provided the opportunity to conceive whenever the current simulation day (currentDATE) is equal to the fertile estrus (FESTR) plus one. If she is not at least 56 weeks of age, she is considered to be ineligible to breed but is not culled as open.

Before any cow can be bred, the bull or artificial insemination technician must be able to detect whether or not the cow is in estrus. If cows are artificially inseminated, the ability to detect the estrus (DETEC) represents management. If cows are mated by natural service, the ability to detect the estrus (DETEC) represents the bull’s libido, its ability to mate with the cow, the bull:cow ratio, pasture terrain, as well as all other environmental conditions (Azzam et al., 1990b). In the BreedFemales function, the probability of not detecting the estrus (NODEC) is set at five percent. An estrus is detected then whenever the random probability of detection (MCDET) selected from a uniform distribution is greater than or equal to the set probability (.05) of not detecting the estrus.

At the time that the estrus is detected, the cow’s record structure counter variable SERVI, the number of
services (detected estruses) occurring during each breeding season, and a program counter called Servi, the total number of services (detected estruses) for all cows annually, are advanced. The first allows the actual number of services per conception for any particular cow to be tracked on DumpList while the latter allows the average number of services per conception (AvServConcp) for the herd to be calculated in ReproStats.

As long as the estrus is detected, the cow or heifer will become pregnant at that estrus if the random probability of pregnancy selected from a uniform distribution is less than or equal to the Bernoulli probability of conception (CONCV) occurring. If pregnancy occurs, the cow’s record structure variable PREG indicating that she is pregnant is set to one. Another record structure variable called CNDAT that indicates the date of conception (CNDAT) is set to the date of the fertile estrus (FESTR). A third record structure variable called PGEST that indicates the potential length of the gestation period for this particular pregnancy is drawn from a normal distribution with a mean and standard deviation equal to the values of the cow’s record structure variables PGAVG and PGSD, respectively. Counters indicating the number of pregnant cows within the herd (PNUM) and the number of cows within the herd that have conceived (CONCE) are also advanced. The function BirthAnimal is called and a fetal
record is created using the breed of the dam and the breed of the herd bull.

For bookkeeping purposes, heifers and cows are counted several times during the breeding season. Cycling Heifers are counted on the first day and at the end of each third of the heifer breeding season while pregnant heifers are counted at the end of each third of the breeding season. The cows are counted on the first day of the regular breeding season. Both cycling and pregnant cows are then counted at the end of each third of the breeding season.


BirthAnimal is used by BreedFemales and SBBreeders to create temporary record structures for unborn calves. Birth-Animal determines if the fetus is a male or female by comparing a Bernoulli variable equal to .50 with a number drawn from a uniform distribution. If the value from the uniform distribution is less than the Bernoulli variable, the sex of the fetus is male. Otherwise it is female. BirthAnimal then calls LastID to obtain an identification number for the fetus. The dam’s record is called and its record structure variable FIDN is assigned the identification number of the fetus while the fetal record structure variable MIDN is assigned the dam’s identification number. The fetal record structure variable for BCS, body condition score, is assigned a starting value of 5 and its
record structure variable PAST indicating if it will eventually become a pasture grazing animal is set to one. All other record structure variables are set to zero including AGE, a variable that indicates the animal’s age. AGE will remain zero until parturition. Breed appropriate values are randomly sampled for the production variables of purebred calves such as birthweight (BWT), potential mature weight (PMWT), potential average daily gain (PADG), and potential mature slaughter weight (PMSWT). The variables for the average length of gestation (PGAVG) and its standard deviation (PGSD) are also assigned at this time. The current weight (CDWT) of the fetus will equal its birth weight until parturition. If the fetus is female and purebred, its production statistics (BWT, PADG, PMWT, PMSWT, PGAVG, AND PGSD) are drawn from a breed appropriate normal distribution. If the fetus is female and crossbred and the simulation’s warm-up period is not finished, its breed, breed history, and crossbred generation number (Fgeneration) will always equal the breed, breed history, and crossbred generation number (Fgeneration) of its dam. This allows the genetic makeup of the breeding herd to remain stable during the warm-up period. After the simulation warm-up period is complete, the production statistics for a female crossbred fetus are chosen from a normal distribution whose mean and standard deviation are computed by combining the production statistics of the dam less heterotic effects with breed
appropriate production statistics for the breed of the sire. After the mean for each production statistic is computed, its standard deviation is calculated by multiplying the mean by an appropriate coefficient of variation from statistics provided by Koots et al. (1994a, b). The crossbred generation number (Fgeneration) is then advanced. BirthAnimal traces the pedigree of each crossbred up to five generations in order to obtain and apply values for direct heterosis and maternal heterosis for birth weight (BWT), potential average daily gain (PADG), and potential mature weight (PMWT). If the fetus is a crossbred male, however, these procedures are used throughout the simulation to obtain beginning values for its production statistics.

4.22. AddAnimal

AddAnimal creates permanent record structures from information provided in the temporary record structures created in CreateHerd, BirthAnimal, and SBBreeders and writes them to the linked list. Each animal’s record structure will then remain on the linked list until it is deleted by the DeleteAnimal routine or sold in the marketing routines.

4.23. LastID

LastID is used to call the last identification number for use in CreateCow, BirthAnimal, and SBBreeders.
4.24. **ABORT**

The ABORT function causes a pregnant female to abort its fetus whenever certain mathematical conditions simulating physiological reality are met. The occurrence of abortion follows a Bernoulli distribution that is dependent upon the female’s age and number of days pregnant. As a cow becomes progressively older, the likelihood of aborting her fetus increases. However as the term of pregnancy increases, the likelihood of aborting declines.

In 1980, Oltenacu et al. described a model in which the probability that a cow will abort its fetus increased by one percent for each year of her age. This follows the findings of Holness et al. (1977) and Kummerfeld et al. (1978) that the percentage of embryonic loss through abortion in cattle increases with the age of the dam. Although Oltenacu et al. (1980) allowed cows in their model to abort at least eight percent of the time, Hawk (1979) reported that the average occurrence of embryonic mortality as a percentage of cattle inseminated averages 15%. Saacke (1998) indicated that probably more than 10% of nonpregnant cattle have aborted on or before the 16th day of pregnancy. Beal et al. (1992) used ultrasound to diagnose pregnancy and verify embryo viability in beef cattle between 25 and 65 days of pregnancy by monitoring fetal heartbeat. The results indicated that 81.25% of the embryonic loss occurred between 25 and 45 days.
of pregnancy while 18.75% occurred between 45 and 65 days. Smith and Stevenson (1995) supported Beal’s conclusions in reporting that approximately 20% of the embryonic losses in their control animals occurred between the 28th day of pregnancy and the 54th day of pregnancy. Hawk (1979) reported that most embryonic loss occurs before nine to 16 days of gestation. In fact, he reported that in Wisconsin there was only a 58% recovery rate of embryos from repeat breeder cows by the 16th day after insemination. Both Beal (1997) and Saacke (1998) reported that two thirds of the embryonic loss occurring by day 45 actually takes place by day 16. Hawk (1979) also reported that fetal mortality occurs only three percent of the time after the 40th day of gestation.

In VBCSM, the probability of losing a fetus is modeled as:

$$\text{ABORT} = 0.115 + (0.01 \times \text{OLD})$$

where,

- **ABORT** = The total likelihood of an abortion.
- **OLD** = The age in years of the pregnant cow.

The probability of an abortion occurring on any day through the 16th day of gestation is modeled as a Bernoulli distribution represented as,

$$p = (\text{ABORT} \times 0.6111) / 16$$

where, \( p \) is the probability of an abortion occurring and .6111 is the proportion of total abortions that occur
through day 16 of pregnancy. The probability of an abortion occurring on any day between the 16th day and the 40th day of gestation is modeled as a Bernoulli distribution represented by:

\[ p = \frac{\text{ABORT} \times 0.3589}{24} \]

After the 40th day of gestation, the probability of an abortion occurring on any day is:

\[ p = \frac{\text{ABORT} \times 0.03}{\text{Gestation Length} - 40} \]

If the Bernoulli probability that is calculated is less than or equal to the random probability that an abortion takes place represented by a number drawn from a uniform distribution, \( u(0, 1) \), the cow aborts. If not the cow remains pregnant.

**4.25. Mortality**

From a large study, Patterson et al. (1987) reported that 6.4% of all calves were lost during the time period between birth and weaning. Rice (1994) reported that 69% of calf losses between birth and weaning occurred within the first 96 hours after birth. Patterson et al. (1987) also found that calf losses differed significantly among dam age and parity classes. Reynolds et al. (1986) reported that survival rate of crossbred calves consistently exceeded that of straightbred calves. Bellows et al. (1987) reported that the largest cause of death in calves was dystocia while death from diseases ranked second in importance.
Mortality calculates all death losses in the herd. In VBCSM, all calf losses occur after parturition. The mortality function utilizes the stochastic variable MCDIE, drawn from a uniform distribution, and the animal’s age to determine if death occurs (Table 4-3).

If the random value of MCDIE is less than the probability of death based upon age and sex, a variable in the animal’s record structure called DIE is advanced to one. Appropriate counters representing the number of dead calves (DNUM), the number of dead unweaned heifers (DNUMH), the number of dead unweaned steers (DNUMS), the number of dead stocker heifers (DNUMHS), the number of dead stocker steers (DNUMSS), the number of dead breeding cows (DNUMF), and the number of replacements that are required (REPLR) are advanced. Appropriate counters representing the number of within herd replacement heifers (NumAvailRepla), the number of stocker heifers available for sale (NSHFC), the number of stocker steers available for sale (NSSTC), the total number of unweaned calves on hand (TNCOH), and the total number of stocker calves on hand (TNSCOH) are decreased. The animal’s record structure variable called AGED that represents its age at death is set to the animal’s current age.

If the animal marked to die is pregnant, the linked list record structure representing its fetus is called and then deleted. The counter representing the number of pregnant cows is decreased. If the animal is lactating, its
calf’s record is called and marked to be sold as an orphan calf while the variables representing the identification number of its calf (CIDN) and the sex of its calf (CSEX) are set to zero. If an orphan calf was previously selected to be a breeding replacement, the counter representing the number of replacements required is advanced while the counter representing the total number of calves on hand is decreased.

If the animal marked to die represents an unweaned calf, its dam’s records are called using the calf’s maternal identification number (MIDN) and the dam’s record structure variables that indicate the calf’s identification number (CIDN), sex (CSEX), and birth date (CBIRTHDATE) are reset to zero. The variables that indicate the dam’s lactation status (LACT), days lactating (LDAY), milk fat content (MCMFC), milk solids content (MCSNF), daily milk yield (YN), and energy contained in her milk (YEN) are also reset to zero and the dam’s record structure variable FWEAN is advanced by one. The counter CowsFailWean indicating the number of cows in the herd that failed to wean a calf during the current year is also advanced by one.

4.26. **SBBreeders**

SBBreeders selects within herd replacements and purchases open and bred replacement heifers. If bred heifers are to be purchased following pregnancy test, VBCSM
creates a record for the heifer based upon the user specifications for breed and purebred or crossbred status. A temporary record structure is created for the bred heifer similar to that discussed for the functions CreateHerd and BirthAnimal except that its age is randomly sampled from a normal distribution with a mean of 570 days and a standard deviation of 35 days.

A value is also stochastically generated from a normal distribution for the bred heifer’s record structure variable DPREG, the number of days since conception by using the stochastic variable MCHOW and the random number generator for a normal distribution, GetStochasticNumber. The mean value used to generate MCHOW is equal to one half the length of the breeding season plus 25 days plus the number of days after the end of the breeding season that pregnancy testing (PTEST) takes place. The standard deviation is calculated by multiplying one half the length of the breeding season by 0.8 and then dividing by 3.9 (the number of standard deviations in a normal distribution table). This is represented by,

\[
MCHOW = \text{normal distribution (STANDARD DEVIATION, MEAN)}
\]

\[
\text{STANDARD DEVIATION} = [.8 \times (\text{Length of the Breeding Season} - (\text{Length of the Breeding Season}/2))/3.9]
\]

\[
\text{MEAN} = (\text{Length of the Breeding Season}/2) + 25 + \text{PTEST}
\]

After building the bred heifer’s temporary record, it is added to the linked list by using the function AddAnimal.
Counter variables representing the number of bred heifers purchased (PHEF), the number of breeding cows in the herd (BreedingCows), and the number of pregnant cows in the herd (PNUM) are then increased. Counter variables representing the number of replacement heifers required (REPLR) and the number of cows testing not pregnant (TSTOP) are decreased. The bred heifer’s conception date (CNDAT) is calculated by subtracting the number of days since conception (DPREG) from the current simulation date. If the conception date is negative, a value of 365 is added to it. A record structure is then created for its fetus using BirthAnimal and added to the linked list using AddAnimal.

After creating the fetal record, its record structure variables representing its identification number, its anticipated birth weight, and its sex are put into the dam’s record structure variables for FIDN, CBWT, and CSEX, respectively. A value representing the current weight of the dam’s gravid uterus in kilograms is then calculated. Then, new values are calculated for the dam’s shrunk body weight in kilograms (SBWTK), its current weight in kilograms (CDWTK), its current weight in pounds (CDWT), its empty body weight in kilograms (EBWTK), its equivalent shrunk body weight in kilograms, and its equivalent empty body weight in kilograms.

The process is essentially the same if nonpregnant replacements are to be purchased upon pregnancy testing or
to fill voids in the breeding herd at other times or in the event that pregnancy testing is not done. The age of the nonpregnant replacement heifer is, however, drawn from a normal distribution with a mean value of 254 days and a standard deviation of 27 days. After the temporary record structure is built, it is also added to the linked list by using the function, AddAnimal. Counters that indicate the number of nonpregnant heifers purchased (POHEF) and breeding cows (BreedingCows) are increased while the counter that indicates the number of replacement heifers required (REPLR) is decreased.

Whenever breeding herd replacements are to be selected from within the herd before purchasing open replacements, the SBBreeders function uses the GetSex function to search through the weanling heifers for replacements. When one is found, its animal record structure variable REPLA representing an available replacement is set to one. After selecting each replacement, counter variables representing the number of replacement heifers required (REPLR) and the number of available replacements within the herd (NumAvailRepla) are decreased. A counter variable indicating the number of replacement heifers selected from within the herd (SELECT) is advanced. The counter variable BreedingReplacement is also increased. The sorting continues as long as either replacement heifers are required or until all animals have been sorted.
4.27. GetSex

GetSex is called in the SBBreeders function to search the herd for potential replacement heifers.

4.28. Wean

Wean collects statistics on calves at the time of weaning. Annual statistics are collected for the total weaning weights of steers, heifers, and all calves; the total numbers of steers, heifers and all calves weaned; the numbers of spring-born calves weaned and the numbers of fall-born calves weaned; and the number of replacement females selected from within the herd. Wean places the animal’s current age into the record structure variable WEAN to record the animal’s age at weaning and places the current date into the record structure variable WEANDATE to indicate the date on which the animal was weaned. It resets the record structure variables for YN, the total amount of milk received during the current day from its dam, and YEN, the total daily energy provided by its dam’s milk for the current day, to zero.

Wean also collects year-end breeding herd statistics. Data collected include the number of cows in each age group from less than 12 months to 14 years of age, and the number of pregnant cows on hand at year end. Wean also collects the combined age of the cows in the breeding herd and the number of nonpregnant cows in each body condition score
category from one to nine on the last day of the last breeding season. The totals are transferred into the marketing module and summarized in the year end budget reports. The combined age is used to calculate and report the average age of the breeding herd in the year end budget report.

4.29. ChangeSex

ChangeSex is used to change the sex designation of non-breeding herd replacement calves to stocker designations at weaning and to change the sex designations of replacement female calves to breeding cows at puberty. At weaning, ChangeSex changes the sex designation of non-replacement weanling heifers from a zero to a four, steers from a two to a five, and breeding herd replacement heifers from zero to a three. After the calf’s sex designation is changed, its current age is stored in its animal record structure variable WEAN and added to either TSWEANAGE, the total age of all calves weaned from the spring calving season, or to TFWEANAGE, the total age of all calves weaned from the fall calving season. The calf’s potential average daily gain (PADG) is also adjusted to no longer reflect maternal heterosis for average daily gain. Lastly, ChangeSex calls its dam and sets her record structure variables for CIDN (calf identification number), CBIRTHDATE (calf birthday), LACT (lactating status), LDAY (consecutive lactating days),
MCMFC (milk fat content), MCSNF (non-fat solids content), YN (current daily milk yield), and YEN (energy in the current daily milk yield) to zero.

4.30. ReproStats

Reprostats collects statistics at the end of each simulation year. It collects and reports annual statistics for the calving rate, the weaning rate, the culling rate, the conception rate, the average birth weight by sex and overall, the average weaning weight by sex and overall, the average length of gestation, and the average number of services per conception. It also collects statistics for the number of eligible breeding cows, the average calving date, the average weaning age of calves born during the spring calving season, and the average weaning age of calves born during the fall calving season.

4.31. DeadCattle

DeadCattle is used to call the function, DeleteAnimal. Before DeleteAnimal is called, a counter called BreedingCows indicating the current number of breeding cows in the herd is decreased. This happens whenever DIE, a variable indicating the animal is marked for death, equals one and AGE, the current age of the animal, equals AGED, its age at death, and the cow is designated as a non-cull breeding cow or a replacement heifer. If the animal is a breeding cow marked for culling after lactation is finished, the counter
for the number of lactating cull cows remaining in the herd, CulledCows, is also decreased. As soon as the current age of any animal marked to die becomes greater than its age at death, it is deleted from the herd by calling its identification number. This is represented as,

\[
\text{If (DIE=1) and (AGE>AGED) then DeleteAnimal(ID)}
\]

4.32. **DeleteAnimal**

DeleteAnimal is a standard C++ function that is called by other functions to delete an animal’s record from the linked list after its death, abortion, or sale.

4.33. **DumpList**

DumpList is a standard function that can be activated during the debugging process to print the record structure variables of each animal to a file. These can be printed daily, weekly, monthly, or annually.

4.34. **ResetCowVariables**

ResetCowVariables resets the general program variables and counters that are used in collecting and reporting annual statistics to zero on the 365\(^{th}\) day of each simulation year.

4.35. **UseYearOption**

UseYearOption is used to give the user the option to change the breed of the herd sire at the end of year one and
year two.

4.36. **DeleteHerd**

DeleteHerd is used to delete the herd when the simulation is completed to free-up computer memory.

4.37. **ResetCowConCntr**

ResetCowConCntr is used to reset counters whenever the simulation program is rerun before completely exiting the program.
Literature Cited


Laster, Danny B. 1974. Factors affecting pelvic size and


Table 4-1. Average slaughter weights and their standard deviations for 12 cattle breeds common to Virginia (Cundiff et al., 1981; USMARC, 1990; Koots et al., 1994a, b).

<table>
<thead>
<tr>
<th>Breed</th>
<th>Average Slaughter Weight (Lbs)</th>
<th>σ (Lbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angus</td>
<td>969</td>
<td>26.000</td>
</tr>
<tr>
<td>Hereford</td>
<td>1038</td>
<td>27.850</td>
</tr>
<tr>
<td>Polled Hereford</td>
<td>1038</td>
<td>27.850</td>
</tr>
<tr>
<td>Shorthorn</td>
<td>988</td>
<td>26.510</td>
</tr>
<tr>
<td>Charolais</td>
<td>1238</td>
<td>33.218</td>
</tr>
<tr>
<td>Chianina</td>
<td>1350</td>
<td>36.223</td>
</tr>
<tr>
<td>Gelbvieh</td>
<td>1175</td>
<td>31.527</td>
</tr>
<tr>
<td>Limousin</td>
<td>1113</td>
<td>29.864</td>
</tr>
<tr>
<td>Maine Anjou</td>
<td>1238</td>
<td>33.218</td>
</tr>
<tr>
<td>Pinzgauer</td>
<td>1075</td>
<td>28.844</td>
</tr>
<tr>
<td>Simmental</td>
<td>1213</td>
<td>32.547</td>
</tr>
<tr>
<td>Tarentaise</td>
<td>1062</td>
<td>28.495</td>
</tr>
</tbody>
</table>
Table 4-2. Variables included in the Dystocia Function (Laster et al., 1973; Meadows et al., 1994).

<table>
<thead>
<tr>
<th>Age of Dam &amp; Sex of Calf</th>
<th>Variable Name</th>
<th>Type or Distribution</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 year, Male</td>
<td>DYPerM2</td>
<td>Bernoulli</td>
<td>0.4349</td>
</tr>
<tr>
<td>2 year, Female</td>
<td>DYPerF2</td>
<td>Bernoulli</td>
<td>0.2840</td>
</tr>
<tr>
<td>3 year, Male</td>
<td>DYPerM3</td>
<td>Bernoulli</td>
<td>0.1372</td>
</tr>
<tr>
<td>3 year, Female</td>
<td>DYPerF3</td>
<td>Bernoulli</td>
<td>0.0592</td>
</tr>
<tr>
<td>4 year +, Male</td>
<td>DYPerM4</td>
<td>Bernoulli</td>
<td>0.0457</td>
</tr>
<tr>
<td>4 year +, Female</td>
<td>DYPerF4</td>
<td>Bernoulli</td>
<td>0.02621</td>
</tr>
<tr>
<td></td>
<td>MCDYSF</td>
<td>Uniform</td>
<td>U(0,1)</td>
</tr>
</tbody>
</table>
Table 4-3. The probability of death occurring on any day based upon age of the beef animal (Patterson et al., 1987).

<table>
<thead>
<tr>
<th>SEX</th>
<th>AGE</th>
<th>PROBABILITY OF DEATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or 2</td>
<td>1 day</td>
<td>0.038500000</td>
</tr>
<tr>
<td>0 or 2</td>
<td>2 day</td>
<td>0.003618000</td>
</tr>
<tr>
<td>0 or 2</td>
<td>3 day</td>
<td>0.001943800</td>
</tr>
<tr>
<td>0 or 2</td>
<td>4 day</td>
<td>0.002010000</td>
</tr>
<tr>
<td>0 or 2</td>
<td>5 day</td>
<td>0.001340000</td>
</tr>
<tr>
<td>0 or 2</td>
<td>6 day</td>
<td>0.000804000</td>
</tr>
<tr>
<td>0 or 2</td>
<td>&gt; 6 days &amp; &lt;= 9 days</td>
<td>0.001055250</td>
</tr>
<tr>
<td>0 or 2</td>
<td>10 days</td>
<td>0.000737000</td>
</tr>
<tr>
<td>0 or 2</td>
<td>&gt; 10 days &amp; &lt;= 41 days</td>
<td>0.000216600</td>
</tr>
<tr>
<td>0 or 2</td>
<td>&gt; 41 days &amp; &lt;= 101 days</td>
<td>0.000063650</td>
</tr>
<tr>
<td>0 or 2</td>
<td>&gt; 101 days</td>
<td>0.000341440</td>
</tr>
<tr>
<td>3</td>
<td>&gt; Puberty</td>
<td>0.000041096</td>
</tr>
<tr>
<td>4 or 5</td>
<td>&gt; Weaning &amp; &lt;= 3 years</td>
<td>0.000054795</td>
</tr>
</tbody>
</table>

*0 = Unweaned heifer; 2 = Unweaned steer; 3 = Breeding cow; 4 = Stocker heifer; 5 = Stocker steer.
Chapter 5  Forage and Nutrition Modules

5.1. Introduction

5.2. FORAGE

5.2.1. UseInput5
5.2.2. UseInput9
5.2.3. UseInput10
5.2.4. UseInput6
5.2.5. MainForage
5.2.6. ResetForageConCntr
5.2.7. ResetForageVariables
5.2.8. Temperature

5.3. NUTRITION

5.3.1. UseInput7
5.3.2. UseInput8
5.3.3. MainNutrition
5.3.4. DWGAIN
5.3.6. FeedUsage
5.3.7. ResetNutritionVariables

Literature Cited
Chapter 5

Forage and Nutrition Modules

Abstract: The forage (Forage) and the nutrition modules (Nutrition) in the Virginia Beef Cattle Simulation Model (VBCSM) contain interacting functions that simulate the growth of forage in Virginia and its consumption and effect on animal growth. Although most variables are deterministic and depend upon user-specifications, several in these modules are stochastic.

5.1. Introduction

The purpose of this chapter is to describe the forage (Forage) and nutrition (Nutrition) modules of the Virginia Beef Cattle Simulation Model (VBCSM). This will be accomplished by discussing each major Forage and Nutrition function that impacts the VBCSM model and the concepts justifying them in the order in which they are executed by VBCSM.

5.2. FORAGE

5.2.1. UseInput5

UseInput5 stores the information gathered in the first page of dialog (Figure A-8, APPENDIX A) under the “Input Forage Values” prompt of the “Main Menu.” This function stores the regional location of the farm in Virginia (Tidewater, Piedmont, or Mountain) and the type of grazing system in use (continuous grazing, simple rotation, or a
complex rotational system). Other information gathered concerns the erosion conditions (severe, moderate, light, or non-existent), the pasture maintenance or the level of fertilization (low, average, or high), and the amount of weed infestation (severe, moderate, light, or non-existent) on the farm. It also stores whether the general terrain of the farm is level or hilly and its average slope. The user provides information concerning the current rental rates per acre of land in the area, the farm operator’s responsibility for maintaining the farm in an appropriate condition, and the estimated sales value and cost per ton of producing hay.

5.2.2. UseInput9

UseInput9 stores the information gathered in the second page of dialog (Figure A-9, APPENDIX A) under the “Input Forage Values” prompt of the “Main Menu.” It allows the user to input the number of pastures on the farm and then specify the number of acres in each pasture used for permanent grazing and the number of acres used for hay production. If any part of the pasture is used for hay production, the beginning and ending dates that cattle will be grazed on it must be specified. The user also specifies the number of acres that are considered unusable for grazing in each pasture. Such acreage includes woodlands, farmsteads, and working pens.
5.2.3. **UseInput10**

UseInput10 stores the information gathered in the third page of dialog (Figure A-10, APPENDIX A) under the “Input Forage Values” prompt of the “Main Menu.” It allows the user to input the number of extra pastures rented, if any, for use by the stocker cattle and then specify for each rental pasture the number of acres used for grazing and the number of acres used for hay production. If any part of the pasture is used for hay, the beginning and ending dates that stocker cattle will be grazed on this proportion of the rental pasture must be specified. Information concerning the number of acres that are considered unusable for stocker grazing in each rental pasture is also provided.

5.2.4. **UseInput6**

UseInput6 stores the information gathered in the fourth page of dialog (Figure A-11, APPENDIX A) under the “Input Forage Values” prompt of the “Main Menu.” This function stores the forage composition of the farm pastures. The user is allowed to choose from 21 warm and cool season forage species. These forage species and their nutritional data are listed in Table 3-1. Their monthly growth rates are listed in Table 3-2. All grass-clover mixtures are assumed to contain 60 % grass and 40 % clover.

5.2.5. **MainForage**

Main forage contains all routines necessary to compute
the initial forage mass in the pastures and the subsequent daily forage growth and senescence during each month of the calendar year. It also computes the growth of forages to be harvested as hay. It is necessary in any forage-animal grazing system for animals and pastures to interact (Whelan et al., 1984). MainForage interacts with MainNutrition to simulate the animal consumption of each day’s available forage. Such effects are functions of forage growth, senescence, and the unconsumed portion of the initial daily plant mass. In VBCSM, the quantity of the forages available for livestock grazing responds to animal consumption on a daily time interval. This more closely reflects reality by allowing the results of modifications in management such as variation in stocking rates, the calving season, and the types, breeds and classes of cattle that are grazed to be reflected. Additionally, like reality, it allows rapid changes to take place whenever necessary.

The nutritional values of the forage species in VBCSM are depicted in Table 3-1. Each forage species may grow at a different rate during each month of the year (Table 3-2). This gives rise to many types of forage strategies including pastures composed of both warm and cool season forages. The forage growth rates were derived from those found in North Carolina (Chamblee & Green, 1995) by adjusting them to the shorter growing season found in Virginia (Green, 1998; Petersen, 1998). According to Green (1998) such growth
rates depict a "worst case" scenario commonly found in severely eroded Appalachian farms and are adjusted upward depending upon erosion conditions. The initial standing cover of the pasture must be estimated at the beginning of the VBCSM’s warm-up. It is estimated by comparing the annual growth rate of its forage composition to that of Fescue and then adjusting it by a multiplicative factor of 2750 pounds per acre which is the estimated initial plant mass of Stockpiled Fescue in Virginia at the beginning of the year (Petersen, 1998).

Management decisions involving grazing cattle encompass decisions about the stocking densities and the grazing strategy to use. These may be dependent upon the size of the pasture, the amount of erosion, the forage species, the amount of weed infestation, the amount of precipitation, pasture maintenance, and the slope of the pasture. The daily potential forage yield is influenced by each of these factors which are normally both multiplicative and independent of each other (Tao, 1988). Thus the forage yield potential may be represented as,

\[ FYP = \text{Base Growth Rate (Lbs/acre/day)} \times \text{Pasture Size (acres)} \times \text{PADJ} \]

\[ \text{PADJ} = \prod \text{ADJ}_{\text{PRECIP}} \times \text{ADJ}_{\text{NFEST}} \times \text{ADJ}_{\text{PMAIN}} \times \text{ADJ}_{\text{GMANGT}} \times \text{ADJ}_{\text{EROS}} \times \text{ADJ}_{\text{SLOP}} \]
where,

- \( FYP \) = Forage Yield Potential, Lbs/acre/day
- \( PADJ \) = Pasture growth rate adjustment
- \( ADJ_{PREcip} \) = Adjustment for precipitation
- \( ADJ_{NFEST} \) = Adjustment for weed infestation
- \( ADJ_{PMAIN} \) = Adjustment for pasture maintenance
- \( ADJ_{GMANGT} \) = Adjustment for grazing system
- \( ADJ_{EROS} \) = Adjustment for soil erosion
- \( ADJ_{SLOP} \) = Adjustment for pasture slope

It is necessary to understand how the environment, the forage, and the livestock interact in order to effectively balance the demand for forage against forage supplies (Tao, 1988; Tao et al., 1991). Grazing capacity of any pasture is determined by the amount of available forage contained within it which, in turn, depends upon the biological yield of the forage resource (Waller et al., 1986; Tao; 1988). The biological yield of the forage resource depends upon the growth rates. The growth rates of any forage species is directly dependent upon the amount of rainfall (Rayburn, 1998). As rainfall increases, growth rates increase. As rainfall decreases, growth rates decrease. Rainfall is stochastically generated, according to the region of Virginia in which the farm is located, using the VBCSM tools section random number generator for a normally distributed variable of defined mean and standard deviation. The monthly averages and standard deviations were calculated.
using monthly historical data from each weather station located in the appropriate region from 1961 through 1991 (Virginia State Climatology Office, 1998). Historical monthly averages and standard deviations in inches of rainfall for each region in Virginia are shown in Table 5-1. Table 5-2 shows the multiplicative effects of stochastically generated rainfall amounts upon forage growth.

The competition of weeds for available plant nutrients also affects the growth rates of forages. Following the indications of Tao (1988), VBCSM defines weed infestation as,

**Severe:** Large amounts of undesirable vegetation. Desirable grass stands are poor. Pasture is poor in quality. Livestock mostly graze the invading plants.

**Moderate:** Undesirable vegetation affects grass stands and reduces pasture productivity. Desirable grass stands are good where there are no invading plants.

**Light:** Generally small amounts of undesirable vegetation and pasture productivity is little affected. Desirable grass stands are good.
Non-existent: Pasture has negligible amounts of undesirable vegetation. The pasture is excellent in quality.

Weed infestation effects in the VBCSM are depicted in Table 5-3.

Forage production responds to soil fertilization. Liming with dolomitic ground limestone supplies calcium and magnesium to the soil while reducing acidity as well as the solubility of aluminum and other elements that often restrict plant growth (Ball et al., 1991). Lime and phosphate applications are required on most Virginia soils for both grasses and clovers to improve forage quality. Nitrogen is deficient in most Virginia soils and its application increases forage production (Blazer, 1986). Fertilization of Virginia soils improves the mineral balances contained in forages, increases yields, and helps to reduce weeds. VBCSM follows Tao (1988) by assuming that the response to fertilization is constant for each maintenance level. Table 5-4 depicts the multiplicative effects of pasture maintenance used in the VBCSM.

Daily forage senescence as applied to pasture productivity can be defined as the daily loss of ungrazed forages due to livestock trampling, wildlife consumption and trampling, weathering, and insects. Like Tao (1988), the VBCSM assumes that forage senescence is equal to a daily loss of 0.36% of the forage mass that is left over
at the end of the day.

Grazing management is defined through the interactions of the stocking rate and the grazing schedule. When cattle graze a pasture continuously for the whole grazing season, forage yields for that pasture will be lower than when cattle are alternated between different pastures for varying periods of time. Grazing management is defined and categorized using multipliers expressed by Tao (1988) as,

**Continuous Grazing**  A system whereby cattle continuously graze a pasture for the entire grazing season.

**Simple Rotation Grazing**  A system in which the cattle are rotated between grazing areas. There is normally a one to two month occupancy between rotations.

**Complex Rotation Grazing**  A system in which cattle are rotated among grazing areas during the active growth stages of the grass. Each occupancy usually lasts 10 days or less. Each area is grazed at least three times during
the growing season.

Grazing effects are depicted in Table 5-5.

Soil erosion refers to the amount of top soil that is missing. As thickness of the top soil decreases, plant rooting becomes more restricted. The VBCSM also follows the classification of Tao (1988) in defining soil erosion but assigns impacts on yield according to Green (1998) as,

**Severe:** Large amounts of top soil gone. Forages growing in subsoils.

**Moderate:** A thin layer of top soil exists, but plant rooting occurs mainly in the subsoils.

**Light:** Some top soil has eroded. Little restriction of plant rooting.

**Non-existent:** No erosion of top soil. Deep soil provides good conditions for plant rooting.

The effects of erosion are depicted in Table 5-6.

Holechek (1988) reported that in rugged terrain, cattle tend to gather more in the convenient flat areas such as in the bottom of a valley or the top of a ridge or in a riparian zone. As pasture slope increases, cattle graze its forages less and less. The VBCSM follows the recommendation of Holechek (1988) and adjusts forage availability by the capacity of cattle to graze it. Its availability is reduced by one percent for each degree of pasture slope in excess of
In VBCSM, the user is allowed to define up to six pastures for cattle operations plus an additional six rental pastures for the stocker operation, if needed. Each pasture can be used as permanent pasture for grazing, hay, or a combination of grazed pasture and hay. The amount of acreage devoted to livestock grazing may change during the year as cattle are either taken off hay pastures in the spring or are again allowed to graze them after the hay is harvested. For the total farm acreage, the growth of both hay and forage is calculated daily and adjusted according to changes between the size of the grazing unit and the size of the haying unit. This is in addition to adjustments to account for soil erosion, the grazing system, the pasture slope, the weed infestation, the amount of fertilization, the monthly rainfall, and the daily senescence.

5.2.6. ResetForageConCntr

ResetForageConCntr resets the variables that represent the nutritional values of the user-specified forage whenever the simulation is rerun without first exiting it. Such variables include FAME, the metabolizable energy of the forage, FANEM, the net energy available in the forage for maintenance, FANEG, the net energy available in the forage for gain, TDNP, the percentage of total digestible nutrients in the forage, and mainFDM, the forage’s dry matter content.
5.2.7. **ResetForageVariables**

ResetForageVariables resets the general variables and counters in the forage module to zero that are used to collect annual statistics on the $365^{th}$ day of each simulation year.

5.2.8. **Temperature**

Using historical averages and standard deviations by region, Temperature simulates a monthly random temperature (Table 5-7; Virginia State Climatology Office, 1998).

Temperature is used to obtain a current monthly temperature ($T_c$) and a previous monthly temperature ($T_p$) for use in the nutrition module. $T_c$ is used in the nutrition function MainNutrition to adjust the daily dry matter intakes of forages and supplemental feeds for each animal in the simulation. Both $T_c$ and $T_p$ are used in the nutrition function DWGAIN to adjust the base net energy value for maintenance ($NE_M$) for the environmental temperature. Both are obtained by calling the random number generator for normal distributions GetStochasticNumber located in the tool module and inputting the average temperature and its standard deviation for the month in question. This is represented as,

$$T_c (T_p) = \text{GetStochasticNumber(Monthly Regional Temperature Standard Deviation, (Monthly Regional Average Temperature)}$$
5.3. NUTRITION

5.3.1. UseInput7

UseInput7 stores the information gathered in the first page of dialog (Figure A-12, APPENDIX A) under the “Input Nutrition Values” prompt of the “Main Menu.” This function determines from the user if supplemental feeds will be fed to the cattle to make up for any nutritional shortfalls resulting from inadequate supplies of forage available for grazing. It stores the cost per pound and the percentage composition of each ingredient in the feed supplement. The user is allowed to choose from 15 feed ingredients including hays and concentrates (Table 3-3). The user is allowed to define five additional concentrate ingredients by providing their names, costs per pound, and their decimal proportion of the feed supplement. Any hays composed of grass-clover mixtures are assumed to contain 60 % grass and 40 % clover.

5.3.2. UseInput8

UseInput8 stores the information gathered in the second page of dialog (Figure A-13, APPENDIX A) under the “Input Nutrition Values” prompt of the “Main Menu.” It stores the name and the nutritional values of the user defined ingredients. Information stored includes the amount of the metabolizable energy (ME), the net energy available for maintenance (NEₘ), and the net energy available for gain (NE₉) in megacalories per kilogram (Mcal/Kg). Also, it
stores the percentage of dry matter and the percentage of total digestible nutrients contained in the ingredient.

5.3.3. **MainNutrition**

MainNutrition contains all algorithms necessary to compute the dry matter intake of the animal according to the formulas reported by the National Research Council (NRC, 1996). It calculates the total daily intake and apportions it between grazed forages and supplemental feeds if the user has chosen to supplement the cattle. It interacts with MainForage by deducting the amount of forage eaten daily by each animal from the total amount of pasture forage available for grazing. The following algorithms are used to calculate the dry matter intake of forage before adjusting it for supplemental feeds.

For growing calves,

\[
FDMI_{\text{UNADJ}} = \left( \text{SBWT}^{.75} \times (0.2435 \text{ FNE}_M - 0.0466 \text{ FNE}_M^2 - 0.1128) / \text{FNE}_M \right) \times (\text{IMPLANT}) \times (\text{TEMPADJ}) \times (\text{EBFE})
\]

For stocker or yearling calves,

\[
FDMI_{\text{UNADJ}} = \left( \text{SBWT}^{.75} \times (0.2435 \text{ FNE}_M - 0.0466 \text{ FNE}_M^2 - 0.0869) / \text{FNE}_M \right) \times (\text{IMPLANT}) \times (\text{TEMPADJ}) \times (\text{EBFE})
\]

For non-pregnant, beef cows,

\[
FDMI_{\text{UNADJ}} = \left( \text{SBWT}^{.75} \times (0.04997 \text{ FNE}_M^2 + 0.03840) / \text{FNE}_M \right) \times (\text{TEMPADJ}) + 0.2 \text{ YN}
\]
For pregnant beef cows,

\[
FDMI_{UNADJ} = \left( (SBWT^{-.75} \times \left( 0.04997 \times FNE_M^2 + 0.04361 \right) / FNE_M \right) \\
\times (TEMPADJ)) + 0.2 \text{ YN}
\]

where,

\( FDMI_{UNADJ} \) = Unadjusted Forage Dry Matter Intake, Kg.
\( SBWT \) = Shrunken Body Weight, Kg.
\( FNE_M \) = Net Energy Value of the forage diet for Maintenance, Mcal/Kg. For beef cows, if \( FNE_M \) is less than one Mcal/Kg, \( FNE_M \) is set to equal 0.95 Mcal/Kg.
\( IMPLANT \) = Effects of Anabolic Implants (Table 5-8).
\( TEMPADJ \) = Adjustments for Temperature (Table 5-9).
\( EBFE \) = Empty Body Fat Effects (Table 5-10).
\( YN \) = Daily milk yield at current week of lactation, Kg/day.

After calculating the animal’s potential daily dry matter intake of forage, the proportion of the diet supplied by grazing and the proportion supplied by supplemental feeds is calculated. Since the quantity of the available forage affects the feed intake of grazing cattle (NRC, 1996), the amount of the diet supplied through grazing is determined by the amount of the initial daily plant mass in kilograms of dry matter per hectare of grazing pasture. If the amount of the initial daily plant mass that is available is greater than or equal to 1150 kilograms per hectare, the diet will
be fully furnished by grazing. Otherwise, the algorithms used to calculate these proportions in VBCSM are,

\[ \text{GRAZE} = \frac{(.17 \text{ DIPM} - .000074 \times \text{DIPM}^2 + 2.4)}{100} \]

\[ \text{SUPP} = 1 - \text{GRAZE} \]

where,

\text{GRAZE} = \text{The proportion of the total diet supplied by grazing available pasture forages.}

\text{DIPM} = \text{The daily initial forage mass, Kg/DM/Hectare.}

\text{SUPP} = \text{The proportion of the total diet supplied by supplemental feeds.}

The preceding algorithms, however, do not exactly reflect stocking density practiced on most stock farms because too much available forage is left ungrazed. Therefore, modifications that perhaps more accurately reflect reality were made. The amount of supplemental feeds fed to the cattle was adjusted according to the initial amount of available pasture mass per breeding cow. As the amount of forage in their diet is increased, the amount of supplemental feeds fed is reduced. Table 5-11 depicts the multiplicative modifiers used to adjust the proportion of the animal’s diet furnished by supplemental feeds.

If the proportion of the diet supplied by grazing is less than one, forage intake is adjusted to reflect the
addition of supplemental feeds to the animal’s diet as,

\[ FDMI_{\text{ADJ}} = FDMI_{\text{UNADJ}} \times \text{GRAZE} \]

where,

- \( FDMI_{\text{ADJ}} \) = Adjusted Forage Dry Matter Intake, Kg.
- \( FDMI_{\text{UNADJ}} \) = Unadjusted Forage Dry Matter Intake, Kg.
- \( \text{GRAZE} \) = The proportion of the total diet supplied by grazing available pasture forages.

After adjusting the animal’s daily dry matter forage intake, the amount supplied by feeding supplements is calculated. The feed supplement fed to pre- and post-weanling calves may be composed of up to 20 ingredients including five optional user defined concentrates (Table 3-3). The only supplemental feeds available to the breeding cows, however, are hays. The algorithms used to calculate each animal’s daily supplemental feed dry matter intake is, for growing calves,

\[ SDMI_{\text{UNADJ}} = [\text{SBWT}^{.75} \times (0.2435 \times \text{SNE}_M - 0.0466 \times \text{SNE}_M^2 - 0.1128)/ \text{SNE}_M] \times (\text{IMPLANT}) \times (\text{TEMPADJ}) \times (\text{EBFE}) \times (\text{SUPP}) \]

for stocker or yearling calves,

\[ SDMI_{\text{UNADJ}} = [\text{SBWT}^{.75} \times (0.2435 \times \text{SNE}_M - 0.0466 \times \text{FNE}_M^2 - 0.0869)/ \text{SNE}_M] \times (\text{IMPLANT}) \times (\text{TEMPADJ}) \times (\text{EBFE}) \times (\text{SUPP}) \]

for non-pregnant beef cows,

\[ SDMI_{\text{UNADJ}} = [\text{SBWT}^{.75} \times (0.04997 \times \text{SNE}_M^2 + 0.03840)/ \text{SNE}_M] \]
* (TEMPADJ) * (SUPP)

for pregnant beef cows,

$$SDMI_{UNADJ} = \left\{ (SBWT^{1.75} \times (0.04997 \times SNE_M^2 + 0.04361) / SNE_M) \\
* (TEMPADJ) \right\} + 0.2 \times YN \times (SUPP)$$

where,

\(SDMI_{UNADJ}\) = Unadjusted Feed Supplement Dry Matter Intake, Kg.

\(SBWT\) = Shrunken Body Weight, Kg.

\(SNE_M\) = Net Energy Value of the feed supplement diet for Maintenance, Mcal/Kg. For beef cows, if \(SNE_M\) is less than one Mcal/Kg, \(SNE_M\) is set equal to 0.95 Mcal/Kg.

\(IMPLANT\) = Effects of Anabolic Implants (Table 5-8).

\(TEMPADJ\) = Adjustments for Temperature (Table 5-9).

\(EBFE\) = Empty Body Fat Effects (Table 5-10).

\(YN\) = Daily milk yield at current week of lactation, Kg/day.

MainNutrition further adjusts the forage intake of each animal if it is being fed ionophores by increasing its feed efficiency by 12%. Finally, the adjusted forage intake for each animal in the herd is deducted from the amount of available pasture forage. This more closely mimics reality by allowing the amount of available pasture forage to continuously change as each animal attempts to meet its nutritional requirements through grazing. Kartchner et al.
(1979) reported that there is no difference in feed intake between calves born in the spring and calves born in the fall, so no adjustments to feed intake are made for different calving seasons.

5.3.4. DWGAIN

An animal’s daily weight gain reflects its propensity to grow and mature in all body parts under constraints of its particular environment (Fitzhugh, 1976). The environment may limit animal productivity through the quantity and quality of available feeds and the effort that is required to locate, consume, and digest them. For example, restricting energy during late gestation may result in a cow that is in thin body condition at calving. This might increase the interval to the postpartum estrus and reduce the likelihood that the cow shows estrus during a restricted breeding season (Richards et al., 1986).

Genetic variation among breeds for biological and economic traits has been well established (MARC, 1990). In fact, beef producers have become more and more aware of the necessity to match genotypes and management to their environmental and economic conditions in order to increase their efficiency of production (Bourdon and Brinks, 1987). When comparing breeds, it is possible to find individuals that are equal in their potential productive performance even though they are from different backgrounds (Pollak,
1984). VBCSM uses breed appropriate performance data (Table 3-4) and genetic composition in combination with NRC (1996) algorithms to model each animal’s average daily gain during the pre- and post weaning periods.

The DWGAIN function calculates the nutrient values of the animal’s dry matter intakes of forage and supplemental feeds calculated in MainNutrition for all classes of cattle except pre-weanling calves. If the animal is a pre-weaning calf, DWGAIN computes its equivalent dry matter intake for a diet that includes forages, supplemental feeds, and its dam’s milk. The algorithm (NRC, 1996) is symbolized as,

\[
CDMIEQ = \frac{(YEN + (PDMI \times NE_M))}{NE_M}
\]

where,

CDMIEQ = The equivalent dry matter intake of a pre-weanling calf, Kg.
YEN = The energy value of a dam’s milk at the current stage of lactation, Mcal.
PDMI = The dry matter consumption of forages and supplemental feeds, Kg.
NE_M = The net energy for maintenance value contained in a diet composed of only forages and supplemental feeds, Mcal/Kg.

VBCSM assigns a value for the thickness of the animal’s hide based upon its breed designation and its current effective hair length depending upon the season of the year.

A body condition score of five is assigned to all
animals when they are created. If the animal is a mature cow, however, body condition score and a value for the effect of previous plane of nutrition on the animal’s net energy value for maintenance requirement (NEₘ) are calculated daily. These values are calculated by comparing the animal’s current shrunk body weight less the weight of its gravid uterus to its shrunk potential mature body weight and assigning a body condition score as shown in Table 5-12. These body condition scores then are used to adjust the cow’s NEₘ. A value for the net energy expended in maintaining the animal in an acclimatized condition (NRC, 1996) is then calculated as,

\[
CNEM = (A1 \times SBWT^{.75} \times BREFF \times LEFF \times FatCompFac) + (A2 \times (20 - Tp))
\]

where,

- \(CNEM\) = The net energy value required for maintenance adjusted for acclimatization, Mcal/Kg.
- \(A1\) = The thermal neutral maintenance requirement, Mcal/day/SBWT^{.75}.
- \(SBWT\) = The animal’s current shrunk body weight, Kg.
- \(BREFF\) = Breed effect on the NEₘ requirement, 1.2 if Simmental, 1.0 for all other breeds.
- \(LEFF\) = Lactation effect on the NEₘ requirement.
requirement, 1.2 if lactating, 1.0 if not.

\[ \text{FatCompFac} = \text{The effect of the previous plane of nutrition on the NE}_m \text{ requirement. It is based upon body condition score.} \]

\[ \text{A2} = \text{The adjustment for the previous month’s temperature, Mcal/day/SBWT}^{75}. \]

\[ \text{Tp} = \text{The previous month’s temperature in degrees Celsius.} \]

A net energy value for maintenance is calculated for the activity effect of animals maintained on pasture (NRC, 1996) as,

\[ \text{CNEMAP} = (.006 \times \text{FDMI}_{\text{ADJ}} \times (.9 - \text{TDNP})) + ((.05\times\text{TERR})/((\text{DIPM}/1000) + 3) \times (\text{CDWTK}/4.184)) \]

where,

\[ \text{CNEMAP} = \text{The animal activity effect on the net energy value for maintenance if the animal is kept on pasture, Mcal/Kg.} \]

\[ \text{FDMI}_{\text{ADJ}} = \text{Adjusted Forage Dry Matter Intake, Kg.} \]

\[ \text{TDNP} = \text{The total digestible nutrient content of the forage composition in the pastures.} \]

\[ \text{TERR} = \text{Indicates the terrain of the grazing pasture: level = 1, hilly = 2.} \]

\[ \text{DIPM} = \text{The Daily initial forage mass,} \]
CDWTK = The animal's current live weight, Kg.

If the animal is a lactating female, its lactational requirements are calculated. These requirements depend upon its peak week of milk production, its peak milk yield, its current day of lactation, the duration of its lactation, the fat content of its milk, and the milk solids (not fat) content of its milk. Its peak week and peak yield of milk production are dependent upon age effects as depicted in Table 5-13 and breed effects as depicted in Table 5-14. The energy value, E, is adjusted for breed effects as shown in Table 5-14. The lactational requirements may be depicted in a series of equations (NRC, 1996) as,

\[
\begin{align*}
KL &= \frac{1}{\text{Peak Week of Lactation}} \\
AL &= \frac{1}{(\text{Peak Yield of Lactation} \times KL \times e)} \\
YN &= \frac{((\text{Current Lactating Day}/7)/ (AL \times e^{(KL \times (\text{Current Lactating Day}/7)})) \times AEF}{\\
E &= ((.092 \times \text{Milk Fat Content}) + (.049 \times \text{Milk Solids (Not Fat) Content}) - .0569) \times BEF \\
YEN &= E \times YN
\end{align*}
\]

where,

KL = An intermediate rate constant.
AL = An intermediate rate constant.
YN = The daily milk yield for lactation,
\[ E = \frac{\text{The energy content of the milk, Mcal}}{\text{(NE}_{\text{M}})/\text{kg.}} \]

\[ \text{AEF} = \text{Age effect (Table 5-13).} \]

\[ \text{BEF} = \text{The breed effect (Table 5-14).} \]

\[ \text{YEN} = \text{The daily energy secretion in milk at the current stage of lactation, Mcal.} \]

Figure 5-1 contains lactation curves for Hereford, Angus, and Gelbvieh cows using the preceding algorithms from NRC (1996).

The pregnant female requires additional energy to support her pregnancy (NRC, 1996). This ensures not only the proper growth and development of the fetus but that the female will be in an adequate body condition to calve, lactate, and rebreed by the beginning of the next breeding season. NRC (1996) equations assume that the nutritional requirements for pregnancy are proportional to the calf’s birth weight. This, in turn, assumes that the factors affecting the calf’s birth weight have a proportion effect upon the dam’s nutrient requirements during pregnancy. The dam’s additional nutritional requirements during pregnancy are calculated as,

\[ \text{CNEMP} = (\text{CBWT}_{\text{K}} \times ((\text{NE}_{\text{M}}/\text{ME})/.13)) \times \]

\[ (.05855 - .0000996 \text{ DPREG}) \times \]

\[ e^{(.03233 \times \text{DPREG} - (.0000275 \times \text{DPREG}^2))} \times 1000 \]

while the daily weight gain in the gravid uterus (Ferrell,
1998) is calculated as,

\[ GUWTKD = \frac{(CBWTK \times (0.3656 - 0.000523 \times DPREG) \times e^{(0.02 \times DPREG) - (0.0000143 \times DPREG)^2})}{1000} \]

where,

- **CNEMP** = The net energy value required for the maintenance of the pregnant female.
- **CBWTK** = The calf’s birth weight, Kg.
- **NE\_M** = The net energy value of the diet (forages and supplemental feeds) available for maintenance, Mcal/Kg.
- **ME** = The metabolizable energy contained in the diet, Mcal/Kg.
- **DPREG** = The cumulative days that the cow has been pregnant.
- **GUWTKD** = The daily weight gain of the gravid uterus during pregnancy.

The various net energy for maintenance requirements are added into an interim total that is used to calculate the net energy value for maintenance due to cold stress. The maintenance requirement for cold stress is illustrated using a series of equations as,

\[ KM = \frac{NE\_M}{ME} \]
\[ ExternInsul = (7.36 - 0.296 \times WIND + 2.55 \times HAIR) \times HIDE \]
\[ Insulate = InternInsul + ExternInsul \]
SurArea = .09 * (SBWTK - GUWTK)\(^{67}\)

MEIntake = PDMI * ME

HeatProd = (MEIntake - RE)/SurArea

LowCritTemp = 39 - (Insulate *

((HeatProd/SurArea) * .85))

CNEMCS = KM * ((SurArea *

(LowCritTemp - Tc))/Insulate)

where,

KM = The ratio of the diet NE\(_M\) to the diet ME.

NE\(_M\) = The net energy value of the diet
(forages and supplemental feeds)
available for maintenance, Mcal/Kg.

ME = The metabolizable energy contained in the diet, Mcal/Kg.

ExternInsul = The animal's external insulation value, \(C^0/M\text{cal/M}^2/\text{Day}\).

WIND = The average annual wind speed in Virginia. It is assumed to be 10 miles per hour.

HAIR = The effective hair length of the animal's coat for insulation: May - October = 0.2, October - May = 0.5.

HIDE = The thickness of the animal's hide:
Thin (dairy and Brahman breeds) =
1, Average (most European breeds) = 2.

Hereford or Hereford crossbreeds = 3.

\[ \text{Insulate} = \text{The animal's total insulation value, } C^0/\text{Mcal/M}^2/\text{Day.} \]

\[ \text{InternInsul} = \text{The animal's tissue (internal) insulation value, } C^0/\text{Mcal/M}^2/\text{Day} \]

(Table 5-15).

\[ \text{SurArea} = \text{The animal's surface area, } \text{M}^2. \]

\[ \text{SBWTK} = \text{The animal’s current shrunk body weight, Kg.} \]

\[ \text{MEIntake} = \text{The animal's total daily metabolizable energy intake, Mcal/day.} \]

\[ \text{PDMI} = \text{Predicted dry matter intake, Kg.} \]

\[ \text{HeatProd} = \text{The animal's daily body heat production, Mcal/day.} \]

\[ \text{LowCritTemp} = \text{The animal's lower critical body temperature, } C^0. \]

\[ \text{CNEMCS} = \text{Net energy value for maintenance required by the animal in cold stress conditions, Mcal/Day.} \]

The amount of dry matter intake required for the animal’s maintenance and the amount of retained energy
available for gain is computed daily. This is depicted, for pre-weaning calves (ionophores not administered) as,

\[
IM = \left(\frac{CNEM + CNEMAP + CNEMCS}{NE_M}\right) \times TcEFF
\]

\[
RE = (CDMIEQ-IM) \times NE_G
\]

And for pre-weanling calves (ionophores administered) as,

\[
IM = \left(\frac{CNEM + CNEMAP + CNEMCS}{(NE_M \times 1.12)}\right) \times TcEFF
\]

\[
RE = (CDMIEQ-IM) \times NE_G
\]

And for breeding cows (Body Condition Score greater than five and ionophores not administered) as,

\[
IM = \left(\frac{CNEM + CNEMAP + CNEMCS + YEN + CNEMP}{NE_M}\right) \times TcEFF
\]

\[
RE = (CPDMI-IM) \times NE_G
\]

And for breeding cows (Body Condition Score of five or less and ionophores not administered) as,

\[
IM = \left(\frac{CNEM + CNEMAP + .525 \times CNEMCS + YEN + CNEMP}{NE_M}\right) \times TcEFF
\]

\[
RE = (CPDMI-IM) \times NE_G
\]

And for breeding cows (Body Condition Score greater than five and ionophores administered) as,

\[
IM = \left(\frac{CNEM + CNEMAP + CNEMCS + YEN + CNEMP}{(NEM \times 1.12)}\right) \times TcEFF
\]

\[
RE = (CPDMI-IM) \times NE_G
\]

And for breeding cows (Body Condition Score of five or less and ionophores administered) as,

\[
IM = \left(\frac{CNEM + CNEMAP + .479 \times CNEMCS + YEN}{(NEM \times 1.12)}\right) \times TcEFF
\]

\[
RE = (CPDMI-IM) \times NE_G
\]
\[ \frac{\text{CNEMP}}{\text{NEM} \times 1.12} \times \text{TcEFF} \]

\[ \text{RE} = (\text{CPDMI} - \text{IM}) \times \text{NE}_G \]

And for stocker calves (ionophores not administered) as,

\[ \text{IM} = \frac{\text{CNEM} + \text{CNEMAP} + \text{CNEMCS}}{\text{NEM}} \times \text{TcEFF} \]

\[ \text{RE} = (\text{SPDMI} - \text{IM}) \times \text{NE}_G \]

for stocker calves (ionophores administered),

\[ \text{IM} = \frac{\text{CNEM} + \text{CNEMAP} + \text{CNEMCS}}{\text{NEM} \times 1.12} \times \text{TcEFF} \]

\[ \text{RE} = (\text{SPDMI} - \text{IM}) \times \text{NE}_G \]

where,

\[ \text{IM} = \text{The dry matter intake required to meet the animal’s daily maintenance requirements, Kg.} \]

\[ \text{CNEM} = \text{The net energy value required for maintenance adjusted for acclimatization, Mcal/Kg.} \]

\[ \text{CNEMAP} = \text{The animal activity effect on the net energy value for maintenance if the animal is kept on pasture, Mcal/Kg.} \]

\[ \text{CNEMCS} = \text{Net energy value for maintenance required by the animal in cold stress conditions, Mcal/Kg. This value was adjusted for breeding cows by a parameter of } 0.525 \text{ if ionophores were not administered and by a} \]
parameter of .479 if ionophores were administered to target a BCS of five for the average breeding female.

YEN = The daily energy secretion in milk at the current stage of lactation, Mcal/Kg.

CNEMP = The net energy value required for the maintenance of the pregnant female, Mcal/Kg.

NE\textsubscript{m} = The net energy value of the diet (forages and supplemental feeds) available for maintenance, Mcal/Kg.

TcEFF = The temperature effect factor, i.e., panting caused by heat stress (Table 5-16).

RE = The retained energy in the daily diet available for gain, Mcal/Kg.

CDMIEQ = The equivalent dry matter intake of a pre-weanling Calf, Kg.

PDMI = The daily dry matter intake (forages and supplemental feeds) of the pre-weanling calf, Kg.

CPDMI = The daily dry matter intake (forages and supplemental feeds) of the breeding cow, Kg.
SPDMI = The daily dry matter intake
(forages and supplemental feeds) of
the post-weanling calf, Kg.

Daily gain in shrunk body weight is computed using the
NRC (1996) algorithms and compared with the animal’s genetic
potential average daily gain (Table 3-4) converted from
pounds into kilograms on a shrunk body weight basis. Both
values are reduced by a multiplicative factor of .95215 if
the animal is a heifer (Van Vleck & Cundiff, 1998). The
daily shrunk body weight gain using the NRC (1996) equations
is computed as,

\[ SBWTG = 13.91 \times RE^{-0.9116} \times EQSBW^{-0.6837} \times SEXDIFF \]

It is compared to the daily shrunk body weight gain
using the animal’s potential average daily gain expressed on
a shrunk body basis computed as,

\[ GPSBWGT = PADG \times 0.453599 \times 0.96 \times SEXDIFF \]

where,

SBWTG = The animal’s daily shrunk body weight gain
in kilograms computed using the NRC (1996)
equations.

GPSBWGT = The animal’s genetic potential average
daily gain in kilograms.

RE = The retained energy value for any
particular day, Mcal/day.

EQSBW = The animal's equivalent shrunk body weight
(an NRC term indicating that the animal's
body weight has been adjusted to a standard NRC reference weight) in kilograms.

\[
\text{SEXXDIFF} = \text{The daily weight gain of a heifer calf compared to a steer calf (Van Vleck & Cundiff, 1998)}
\]

\[
\text{PADG} = \text{The animal's potential average daily gain, Lbs/day (Table 3-4).}
\]

If the daily shrunk body weight gain in kilograms using the NRC (1996) equations adjusted for sex differences is less than the genetic potential shrunk body average daily gain adjusted for sex differences, the NRC (1996) computed gain is used. Otherwise, if the NRC (1996) computed daily shrunk body weight gain is greater than the genetic potential average daily gain expressed on a shrunk body weight, the animal only gains its genetically determined potential average daily shrunk body weight gain in kilograms. This is represented in the VBCSM as,

\[
\text{if (SBWTG} \leq \text{GPSBWTG)}
\]

\[
\text{SBWTK} = \text{SBWTK} + \text{SBWTG}
\]

\[
\text{else}
\]

\[
\text{SBWTK} = \text{SBWTK} + \text{GPSBWTG}
\]

The user has the option to feed the stocker calves for a targeted average daily gain. If this option is exercised, a targeted random average daily gain for each animal is calculated using the user-specified anticipated daily gain as a mean and calculating a standard deviation from it using
a coefficient of variation calculated from the data provided by Koots et al. (1994a, b). The animal is only allowed to gain up to its targeted random average daily gain. Any excess retained energy value as represented by the difference between the shrunk body weight gain calculated using the NRC (1996) equations and the targeted random shrunk body weight gain is converted into a value representing excess supplemental feeds eaten by the stocker calves. This value is deducted from the calculated total amount of supplemental feeds eaten by the stocker calves. The animal’s daily shrunk body weight gain is represented as,

\[
\text{SBWTG} = \text{SBWTG} + \text{Daily Random Targeted GAIN}
\]

After computing the animal’s new shrunk body weight gain in kilograms, this value is converted into the animal’s current daily weight in pounds and kilograms for use in other functions and modules. It is also converted into its current empty body weight in kilograms, its current equivalent empty body weight in kilograms, and its current equivalent shrunk body weight in kilograms for use in the next day’s nutritional routines.

#### 5.3.6. FeedUsage

Feedusage calculates the annual costs of the supplemental feeds. The feed cost for the supplemental feeds purchased for the cow-calf herd is calculated before
the feed cost of the supplemental feeds purchased for the stocker calves. After feed costs are calculated, FeedUsage calculates the net revenues of farm grown hay. This amount represents the difference between the sales price of the hay and its production costs per pound. The values used for sales prices and production costs are obtained through dialog interactions. Hay revenues are first credited against the cost of supplemental feeds in the stocker operation before deducting any excess from the costs of supplemental feeds used for the cow-calf operation. By crediting hay at its net value, a production cost is assigned to farm grown hays that must be absorbed by cattle operations. However, it may or may not be realistic to credit hay revenues against the cost of the supplemental feeds purchased for stocker calves before those purchased for the cow-calf operation since this procedure appears to subsidize the stocker calf operation through the cow-calf operation. Any hay produced by the farm that is in excess of the animal consumption is sold. No carryovers are allowed. FeedUsage also collects statistics on the total annual hay and concentrates consumption by both the cow-calf operation and the stocker calf operation on the 365th day of the year.

5.3.7. ResetNutritionVariables

ResetNutritionVariables resets the general variables
and statistical counters in the nutrition module to zero on the 365\textsuperscript{th} day of each simulation year.
Literature Cited


Green, James T. 1998. Personal communication.


Petersen, P. 1998. Personal communication.


<table>
<thead>
<tr>
<th>MONTH</th>
<th>TIDEWATER AVG. RAINFALL (inches)</th>
<th>TIDEWATER $\sigma$</th>
<th>PIEDMONT AVG. RAINFALL (inches)</th>
<th>PIEDMONT $\sigma$</th>
<th>MOUNTAIN AVG. RAINFALL (inches)</th>
<th>MOUNTAIN $\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>3.59</td>
<td>.383</td>
<td>3.12</td>
<td>.274</td>
<td>2.76</td>
<td>.681</td>
</tr>
<tr>
<td>February</td>
<td>3.38</td>
<td>.350</td>
<td>3.18</td>
<td>.326</td>
<td>2.89</td>
<td>.701</td>
</tr>
<tr>
<td>March</td>
<td>3.86</td>
<td>.278</td>
<td>3.66</td>
<td>.269</td>
<td>3.43</td>
<td>.798</td>
</tr>
<tr>
<td>April</td>
<td>2.99</td>
<td>.235</td>
<td>3.25</td>
<td>.255</td>
<td>3.28</td>
<td>.749</td>
</tr>
<tr>
<td>May</td>
<td>3.91</td>
<td>.389</td>
<td>4.16</td>
<td>.295</td>
<td>3.96</td>
<td>.782</td>
</tr>
<tr>
<td>June</td>
<td>3.75</td>
<td>.423</td>
<td>3.56</td>
<td>.322</td>
<td>3.48</td>
<td>.761</td>
</tr>
<tr>
<td>July</td>
<td>4.51</td>
<td>.598</td>
<td>4.23</td>
<td>.346</td>
<td>4.02</td>
<td>.831</td>
</tr>
<tr>
<td>August</td>
<td>4.51</td>
<td>.609</td>
<td>4.05</td>
<td>.346</td>
<td>3.59</td>
<td>.736</td>
</tr>
<tr>
<td>September</td>
<td>3.85</td>
<td>.476</td>
<td>3.55</td>
<td>.338</td>
<td>3.43</td>
<td>.772</td>
</tr>
<tr>
<td>October</td>
<td>3.29</td>
<td>.365</td>
<td>3.88</td>
<td>.434</td>
<td>3.56</td>
<td>.794</td>
</tr>
<tr>
<td>November</td>
<td>3.17</td>
<td>.228</td>
<td>3.42</td>
<td>.337</td>
<td>3.16</td>
<td>.762</td>
</tr>
<tr>
<td>December</td>
<td>3.31</td>
<td>.244</td>
<td>3.16</td>
<td>.181</td>
<td>2.90</td>
<td>.673</td>
</tr>
</tbody>
</table>
Table 5-2. The Multiplicative Effect Of Rainfall Upon Forage Growth In The Virginia Beef Cattle Simulation Model (Rayburn, 1998; Tao, 1988).

<table>
<thead>
<tr>
<th>σ UNITS OF RAINFALL</th>
<th>MULTIPLICATIVE EFFECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater than 3</td>
<td>1.57</td>
</tr>
<tr>
<td>More than + 2</td>
<td>1.38</td>
</tr>
<tr>
<td>Less than or equal to + 3</td>
<td>1.38</td>
</tr>
<tr>
<td>More than + 1</td>
<td>1.19</td>
</tr>
<tr>
<td>Less than or equal to + 2</td>
<td>1.19</td>
</tr>
<tr>
<td>Less than or equal to + 1</td>
<td>1.00</td>
</tr>
<tr>
<td>More than or equal to - 1</td>
<td>1.00</td>
</tr>
<tr>
<td>Less than -1</td>
<td>0.81</td>
</tr>
<tr>
<td>More than or equal to -2</td>
<td>0.81</td>
</tr>
<tr>
<td>Less than -2</td>
<td>0.62</td>
</tr>
<tr>
<td>More than or equal to - 3</td>
<td>0.62</td>
</tr>
<tr>
<td>Less than -3</td>
<td>0.43</td>
</tr>
</tbody>
</table>
Table 5-3. Multiplicative Effect Of Weed Infestation In The Virginia Beef Cattle Simulation Model (Tao, 1988).

<table>
<thead>
<tr>
<th>WEED INFESTATION</th>
<th>MULTIPLICATIVE EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severe</td>
<td>0.30</td>
</tr>
<tr>
<td>Moderate</td>
<td>0.70</td>
</tr>
<tr>
<td>Light</td>
<td>0.90</td>
</tr>
<tr>
<td>Non-existent</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Table 5-4. Multiplicative Effect Of Pasture Maintenance In The Virginia Beef Cattle Simulation Model (Tao, 1988).

<table>
<thead>
<tr>
<th>PASTURE MAINTENANCE</th>
<th>MULTIPLICATIVE EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Levels</td>
<td>0.50</td>
</tr>
<tr>
<td>Average Levels</td>
<td>1.00</td>
</tr>
<tr>
<td>High Levels</td>
<td>1.15</td>
</tr>
</tbody>
</table>
Table 5-5. Multiplicative Effect Of Grazing Management In The Virginia Beef Cattle Simulation Model (Tao, 1988).

<table>
<thead>
<tr>
<th>GRAZING SYSTEM</th>
<th>MULTIPLICATIVE EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>0.75</td>
</tr>
<tr>
<td>Simple Rotation</td>
<td>0.90</td>
</tr>
<tr>
<td>Complex Rotation</td>
<td>1.05</td>
</tr>
</tbody>
</table>
Table 5-6. Multiplicative Effect Of Soil Erosion In The Virginia Beef Cattle Simulation Model (Tao, 1988; Green, 1998).

<table>
<thead>
<tr>
<th>SOIL EROSION</th>
<th>MULTIPLICATIVE EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severe</td>
<td>0.833</td>
</tr>
<tr>
<td>Moderate</td>
<td>1.167</td>
</tr>
<tr>
<td>Light</td>
<td>1.500</td>
</tr>
<tr>
<td>Non-existent</td>
<td>1.667</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MONTH</th>
<th>TIDEWATER</th>
<th>PIEDMONT</th>
<th>MOUNTAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AVG. F₀</td>
<td>σ</td>
<td>AVG. F₀</td>
</tr>
<tr>
<td>January</td>
<td>37.17</td>
<td>1.37</td>
<td>34.07</td>
</tr>
<tr>
<td>February</td>
<td>39.74</td>
<td>1.14</td>
<td>37.02</td>
</tr>
<tr>
<td>March</td>
<td>48.11</td>
<td>1.20</td>
<td>46.18</td>
</tr>
<tr>
<td>April</td>
<td>56.86</td>
<td>1.06</td>
<td>55.27</td>
</tr>
<tr>
<td>May</td>
<td>65.91</td>
<td>0.77</td>
<td>64.09</td>
</tr>
<tr>
<td>June</td>
<td>73.76</td>
<td>0.70</td>
<td>72.00</td>
</tr>
<tr>
<td>July</td>
<td>77.83</td>
<td>0.66</td>
<td>75.90</td>
</tr>
<tr>
<td>August</td>
<td>76.73</td>
<td>0.74</td>
<td>74.62</td>
</tr>
<tr>
<td>September</td>
<td>70.75</td>
<td>1.11</td>
<td>68.04</td>
</tr>
<tr>
<td>October</td>
<td>59.95</td>
<td>1.43</td>
<td>56.80</td>
</tr>
<tr>
<td>November</td>
<td>51.05</td>
<td>1.33</td>
<td>47.70</td>
</tr>
<tr>
<td>December</td>
<td>41.68</td>
<td>1.52</td>
<td>37.96</td>
</tr>
</tbody>
</table>
Table 5-8. Multiplicative Effects Of Using Or Not Using Anabolic Implants On Determining The Amount Of Dry Matter Intake In The Virginia Beef Cattle Simulation Model (NRC, 1996).

<table>
<thead>
<tr>
<th>IMPLANT CATTLE</th>
<th>MULTIPLICATIVE EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>1.00</td>
</tr>
<tr>
<td>NO</td>
<td>0.94</td>
</tr>
<tr>
<td>TEMPERATURE ($F^\circ$)</td>
<td>MULTIPLICATIVE EFFECT</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>77$^\circ$ or more</td>
<td>0.90</td>
</tr>
<tr>
<td>59$^\circ$ to 77$^\circ$</td>
<td>1.00</td>
</tr>
<tr>
<td>41$^\circ$ to 59$^\circ$</td>
<td>1.03</td>
</tr>
<tr>
<td>23$^\circ$ to 41$^\circ$</td>
<td>1.05</td>
</tr>
<tr>
<td>5$^\circ$ to 23$^\circ$</td>
<td>1.16</td>
</tr>
</tbody>
</table>
Table 5-10. Multiplicative Empty Body Fat Effects On Determining The Amount Of Dry Matter Intake In The Virginia Beef Cattle Simulation Model (NRC, 1996).

<table>
<thead>
<tr>
<th>EQUILIVANT EMPTY BODY WEIGHT (KG)</th>
<th>FAT COMPOSITION (%)</th>
<th>MULTIPLICATIVE EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>501 - 550</td>
<td>31.5</td>
<td>0.73</td>
</tr>
<tr>
<td>451 - 500</td>
<td>29.0</td>
<td>0.82</td>
</tr>
<tr>
<td>401 - 450</td>
<td>26.5</td>
<td>0.90</td>
</tr>
<tr>
<td>351 - 400</td>
<td>23.8</td>
<td>0.97</td>
</tr>
<tr>
<td>350 or Less</td>
<td>21.3</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Table 5-11. Multiplicative Modifiers Used To Adjust The Proportion Of The Animal Diet Furnished By Supplemental Feeds Relative To The Initial Daily Forage Mass Available (Kgs/DM/Cow) In The Virginia Beef Cattle Simulation Model.

<table>
<thead>
<tr>
<th>INITIAL DAILY PASTURE MASS (KGS/COW)</th>
<th>MULTIPLIERS USED FOR ADJUSTING THE PROPORTION OF THE DIET FURNISHED BY SUPPLEMENTAL FEEDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 50</td>
<td>1.00</td>
</tr>
<tr>
<td>50 - 100</td>
<td>0.80</td>
</tr>
<tr>
<td>100 - 150</td>
<td>0.60</td>
</tr>
<tr>
<td>150 - 200</td>
<td>0.40</td>
</tr>
<tr>
<td>200 - 250</td>
<td>0.20</td>
</tr>
<tr>
<td>Over 250</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table 5-12. The Effect Of Shrunk Body Weight On Body Condition Score And Net Energy For Maintenance (NRC, 1996).

<table>
<thead>
<tr>
<th>(Shrunk Body Weight)/(Shrunk Potential Mature Weight)</th>
<th>BODY CONDITION SCORE</th>
<th>PREVIOUS NUTRITIONAL EFFECT ON NE&lt;sub&gt;m&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>79% or less</td>
<td>1</td>
<td>0.80</td>
</tr>
<tr>
<td>79% - 83%</td>
<td>2</td>
<td>0.85</td>
</tr>
<tr>
<td>83% - 90%</td>
<td>3</td>
<td>0.90</td>
</tr>
<tr>
<td>90% - 96%</td>
<td>4</td>
<td>0.95</td>
</tr>
<tr>
<td>96% - 104%</td>
<td>5</td>
<td>1.00</td>
</tr>
<tr>
<td>104% - 113%</td>
<td>6</td>
<td>1.05</td>
</tr>
<tr>
<td>113% - 124%</td>
<td>7</td>
<td>1.10</td>
</tr>
<tr>
<td>124% - 137%</td>
<td>8</td>
<td>1.15</td>
</tr>
<tr>
<td>137% or higher</td>
<td>9</td>
<td>1.20</td>
</tr>
</tbody>
</table>
Table 5-13. Age Effects Upon Milk Yield In The Virginia Beef Cattle Simulation Model (NRC, 1996).

<table>
<thead>
<tr>
<th>AGE (YEARS)</th>
<th>MULTIPLIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 – 3</td>
<td>0.74</td>
</tr>
<tr>
<td>3 – 4</td>
<td>0.88</td>
</tr>
<tr>
<td>&gt; 4</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Table 5-14. The Peak Week Of Lactation And The Milk Energy Multiplier Used To Adjust The Energy Value To The Estimated Volume Of Milk Produced By Each Breed In The Virginia Beef Cattle Simulation Model (NRC, 1996; Jenkins & Ferrell, 1992; MARC, 1990).

<table>
<thead>
<tr>
<th>BREED OR PRIMARY BREED OF PUREBRED AND CROSSBRED COWS</th>
<th>PEAK LACTATION WEEK</th>
<th>PEAK LACTATION YIELD (KGS)</th>
<th>ENERGY EFFECT MULTIPLIER INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angus</td>
<td>10.4</td>
<td>8.0</td>
<td>1.00</td>
</tr>
<tr>
<td>Hereford</td>
<td>8.8</td>
<td>7.0</td>
<td>1.00</td>
</tr>
<tr>
<td>Polled Hereford</td>
<td>8.8</td>
<td>7.0</td>
<td>1.00</td>
</tr>
<tr>
<td>Shorthorn</td>
<td>10.4</td>
<td>8.5</td>
<td>1.00</td>
</tr>
<tr>
<td>Charolais</td>
<td>9.5</td>
<td>9.0</td>
<td>0.94</td>
</tr>
<tr>
<td>Chianina</td>
<td>8.0</td>
<td>6.0</td>
<td>0.91</td>
</tr>
<tr>
<td>Gelbvieh</td>
<td>10.0</td>
<td>11.5</td>
<td>1.27</td>
</tr>
<tr>
<td>Limousin</td>
<td>8.8</td>
<td>9.0</td>
<td>0.91</td>
</tr>
<tr>
<td>Maine Anjou</td>
<td>8.8</td>
<td>9.0</td>
<td>0.98</td>
</tr>
<tr>
<td>Pinzgauer</td>
<td>9.6</td>
<td>11.0</td>
<td>1.11</td>
</tr>
<tr>
<td>Simmental</td>
<td>9.6</td>
<td>12.0</td>
<td>1.33</td>
</tr>
<tr>
<td>Tarentaise</td>
<td>10.4</td>
<td>9.0</td>
<td>1.10</td>
</tr>
</tbody>
</table>
Figure 5-1. Lactation Curves Based Upon The Peak Week Of Lactation For Lactating Hereford, Angus, and Gelbvieh Cows In The Virginia Beef Cattle Simulation Model (NRC, 1996; Jenkins & Ferrell, 1992; MARC, 1990).
Table 5-15. Internal Insulation Factors Used In The Virginia Beef Cattle Simulation Model NRC, 1996).

<table>
<thead>
<tr>
<th>AGE</th>
<th>INTERNAL INSULATION FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 30 days</td>
<td>2.5</td>
</tr>
<tr>
<td>30 days – 1 year</td>
<td>6.5</td>
</tr>
<tr>
<td>1 year – 2 years</td>
<td>5.1875 + .3125 BCS*</td>
</tr>
<tr>
<td>More than 2 years</td>
<td>5.25 + .75 BCS</td>
</tr>
</tbody>
</table>

*BCS = Body Condition Score
Table 5-16. The Effect Of Panting Caused By Heat Stress On The Daily Dry Matter Intake Required To Meet Beef Cattle Maintenance Requirements (NRC, 1996)

<table>
<thead>
<tr>
<th>CURRENT TEMPERATURE (°C)</th>
<th>MULTIPLIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 25°</td>
<td>1.00</td>
</tr>
<tr>
<td>25° –30°</td>
<td>1.07</td>
</tr>
<tr>
<td>More than 30°</td>
<td>1.18</td>
</tr>
</tbody>
</table>
Chapter 6  Marketing Module

6.1.  Introduction

6.2. Marketing Alternatives in Virginia

   6.2.1.  Marketing Methods
   6.2.2.  Auction Marketing
   6.2.3.  Direct Marketing
   6.2.4.  Electronic Marketing

   6.2.4.1.  Virginia Tel-O-Market Auction

   6.2.5.  Futures Markets

6.3.  UseInput3

6.4.  UseInput4

6.5.  PurchaseLivestock

6.6.  SellWeanlings

6.7.  SellOrphans

6.8.  SellCullCows

6.9.  SellStockerCalves

6.10. BudgetReport

6.11. ResetMarketVariables

Literature Cited
Chapter 6

Marketing Module

Abstract: The marketing module (Market) in the Virginia Beef Cattle Simulation Model (VBCSM) contains functions that interact with the forage, nutrition, and reproduction and life events modules to buy and sell cattle and provide year end financial reports and herd summaries. Income statements are provided for the cow-calf operation, while partial budgets are provided for the stocker calf operation to determine its profitability in the farm business.

6.1. Introduction

The purpose of this chapter is to describe the marketing (Market) module of the Virginia Beef Cattle Simulation Model (VBCSM). This will be accomplished by first discussing the primary marketing alternatives for beef cattle in Virginia. Next, each major Market function that impacts the VBCSM model and the concepts justifying them will be discussed in the order in which they are executed by VBCSM.

6.2. Marketing Alternatives in Virginia

6.2.1. Marketing Methods

To maximize their returns from beef cattle ventures and to sell their output efficiently, producers should evaluate all marketing options (Yaeger & Greer, 1977). There are several marketing options available to cattlemen. These
options include terminal or central markets, auction markets, electronic markets, direct sales, country commission firms, and producer-owned marketing cooperatives. Futures markets can be used to protect against adverse price movements.

6.2.2. Auction Marketing

Auction markets are small (usually one owner) regional, public markets that are similar to terminal markets (Lesser, 1993). They provide a convenient, federally supervised, local selling and assembly point for cattle. Their market areas are usually within a 40 to 50 mile radius. Auctions normally take place on a specified sales day each week. Selling is by visual inspection. The process is regulated and completely open to public view. This guarantees that all buyers present will have an equal opportunity to bid on any cattle for sale and that buyers and sellers are treated fairly (Davis, 1988). There is a ready market for any animal because it is likely to be sold at some price regardless of its condition (Davis, 1988). The producer pays the cost of transportation to market, yardage, special services, and commission fees (Grinnell, 1973). Marketing costs are not set and will vary from auction market to auction market (Davis, 1988). The seller has no direct input into the pricing process, although many auctions engage in price support activities to encourage producers to consign live-
stock to them (Grinnell, 1973).

6.2.3. Direct Marketing

The direct sales method consists of a producer showing his cattle to a buyer that represents a meat packer, a feedlot, or another producer and negotiating with him to arrive at a private treaty sales price and delivery terms (Lesser, 1993). Negotiation continues until a sale is made or the cattle are sold in some other manner. Selling takes place at a farm, feedlot, or collection yard. Only one buyer at a time may inspect and negotiate on the cattle, and others must wait for their turn and it comes only if the producer rejects the previous buyer's final offer. The producer's direct cost is the time spent in showing his cattle and negotiating their sale.

This method is becoming more and more important as buyers look for cattle that have known backgrounds (Davis, 1988). The producer is better able to inform the buyer of the positive features of his cattle and has unlimited negotiation powers, but he must be able to tolerate the negotiation process. Producers have the opportunity to be paid price premiums for their cattle if they can get the buyer's confidence. They must, however, keep up with the market conditions and be familiar with the conditions for direct sales including pencil shrink, weighing locations, and delivery points to competitively price their cattle.
6.2.4. **Electronic Marketing**

Electronic marketing is a central, simultaneous trading domain for spatially dispersed buyers and sellers and is characterized by some method of electronic communication such as a conference telephone, teletype, computer system, video, or a combination of these items (Russell & Purcell, 1980; Turner et al., 1991). This marketing method has the potential to change the entire way that cattle are bought and sold. Depending upon the electronic communications system used and the commodity marketed, electronic markets can improve market information, increase market efficiency, improve pricing accuracy, increase competition, and improve market accessibility for buyers and sellers (Turner et al., 1991). Since cattle sell by description, this method conceivably increases the number of buyers with market access and reduces marketing costs (Russell & Purcell, 1980). Pricing efficiency may be improved and more responsive short-term shifts in supply and demand take place due to the ease of market entry and exit. Although most producers feel there is not enough competition without at least 10 buyers, a typical auction market normally has a maximum of only five actual buyers (Russell & Purcell, 1980).

Local auction markets remain essential for success in electronic marketing. The auctions are recognized by producers and buyers and can provide assembly, weighing, and
grading facilities to sort the cattle into homogeneous lots. Competition to assemble the cattle will keep charges at reasonable levels and provide the auctions incentives to participate. The marketing manager or the directors normally resolve any disputes arising out of the use of the electronic marketing system. The costs to the cattle producer are transportation to a local collection point, yardage, special services, and commission fees.

6.2.4.1. Virginia Tel-O-Market Auction

The Virginia Tel-O-Market was established in 1973 and sponsored by the Virginia Cattlemen’s Association in affiliation with local feeder cattle sales associations and livestock markets (McKinnon, 1998). It is conducted as a conference telephone auction with consigned cattle remaining on the farm until after they are sold and a delivery date and time has been arranged to deliver them to the local livestock market or weighing station. Consigned calves must weigh between 300 and 700 pounds and yearlings must weigh between 500 and 1000 pounds. Males must be properly castrated and healed when delivered to the sale. Representatives from the Virginia Department of Agriculture and Consumer Services grade the cattle consigned to these sales. Inferior grades, stags, bulls, dwarfs, and sick or defective cattle are not accepted. All cattle must be sold to the highest bidder who must accept delivery on the day
designated by the Virginia Cattlemen’s Association.

6.2.5. Futures Markets

The futures market may be one of the most economical places for a cattle producer to obtain market information (Shonkwiler & Hinckley, 1985). If producers form future market expectations by considering historical trends and current information, the futures price of feeder and slaughter cattle in any contract month will represent the rational part of that expectation. Cattle producers can use the futures markets to hedge (protect) against any adverse price movements that might occur before their cattle are ready to sell. The amount that a cattle producer will receive from hedging will depend upon the relationship between cash prices, futures prices, and the amount of production being hedged (Kenyon et al., 1991). Since basis (the price difference that exists between the cash and the futures market) depends upon the relationship between the futures market price and the cash price, successful hedging depends upon accurately pinpointing that relationship at the time that the hedge is to be removed (Ward, 1987; Kenyon et al., 1991). If there is perfect correlation between cash and futures prices, there will be no risk in hedging (Elam, 1988). In actual practice, however, cash and futures prices are not perfectly correlated and basis risk is present. The net price that a producer will receive from hedging his
cattle is the sum of the existing cash price for live cattle when the hedge is terminated plus a return (positive or negative) on the futures contract trades.

6.3. UseInput3

UseInput3 stores the information gathered in the first page of dialog (Figure A-14, APPENDIX A) under the “Input Market Values” prompt of the “Main Menu.” It stores the type of marketing method (direct marketing or auction barn) used to sell cull cows and orphan calves and the type of marketing method (direct marketing, auction barn, or Tel-O-Market auction) used to sell weanling calves and stocker calves. It also stores the distance to the closest auction barn or gathering pens, and whether or not the calves will be overnighted. This function obtains the annual anticipated selling price per hundred weight for 600 – 700 pound feeder steers, 700 – 800 pound yearling steers, and commercial grade Virginia cows. Additionally, UseInput3 stores whether or not the weanling calves or the stocker calves will be hedged, and if so, the basis, or difference, between Virginia cash prices and the Chicago Mercantile Exchange futures market prices.

6.4. UseInput4

UseInput4 stores the information gathered in the second page of dialog (Figure A-15, APPENDIX A) under the “Input Market Values” prompt of the “Main Menu.” It obtains and
stores values of various costs per unit related to the cow-calf operation and the stocker operation. Costs related to the cow-calf operation that are expressed as a cost per breeding cow are artificial insemination charges, building and fence repairs, livestock supplies, non-crop machinery charges, utility charges, culled cow transportation costs, labor hours, and veterinary medicine and supplies. Some other costs stored for the cow-calf operation are the cost of each implant given to pre-weanling calves, the cost per acre of maintaining the pasture, salt and mineral costs per hundredweight consumed, the cost of transporting each calf to market, and the cost of each hour of labor expended. Also stored for the cow-calf operation are the current federal payroll tax rates and the local interest rates for cow-calf operating loans. Default values obtained from the budgets of the Virginia Cooperative Extension Service (1995) are provided for each cow-calf item.

Costs obtained and stored for the stocker calf operation that are expressed as a cost per stocker calf include building and fence repairs, hours of labor expended, livestock supplies, non-crop machinery costs, insurance charges, market transportation costs, utilities, and veterinary medicine and supplies. Additionally, UseInput4 gathers and stores values for the cost of maintaining each acre of rental pasture used for the stocker herd and the salt and mineral costs per hundredweight consumed. It also
obtains the interest rates charged on operating loans for the stocker calf operation. Default values obtained from the Virginia Cooperative Extension Service (1995) are provided for each stocker calf item.

6.5. **PurchaseLivestock**

Cattle producers can choose to replace culled cows by selecting from the available heifers within the herd, purchasing nonpregnant heifers, or purchasing bred heifers. The selection of any of the aforementioned methods will affect not only the reproductive performance of the herd but also its genetic merit and economic productivity. The decision on which of the three methods to use is normally based upon the cost of replacements and the time value of money as a cattle producer attempts to match the biological type of the cow to the most economical feed conditions (Rodgers, 1972). By purchasing bred heifers, the cattle producer avoids the costs of maintaining open replacements until the start of the next breeding season. However their cost may exceed that of nonpregnant heifers, and selectivity is usually much less.

*PurchaseLivestock* first adjusts the estimated base annual price paid per hundred weight of a 600 – 700 pound feeder steer to an average monthly price using the monthly indexes depicted in Table 3-7. After adjusting the annual base price to a monthly price, the purchase cost of an open
heifer is calculated according to the adjustments for its weight as depicted in Table 3-6 and multiplied by the animal’s weight. It is assumed that all purchased heifers are bought at their net weight. Nonpregnant heifers may be purchased under all culling policies and whenever a cow dies. This is represented as,

\[
\text{if (SEX = Female) and (Source = Purchased) then,}
\]

\[
\text{Purchase Price} = \text{(Base Monthly Price)} \times \text{(Price Adjustment For Weight)} \times \text{(Current Weight)}
\]

If, however, the animal that is purchased is a bred heifer, the price is further adjusted by using a 1.12 multiplier (Table 3-8) signifying the difference in prices per hundredweight between bred heifers and 600 - 700 pound feeder steers. However, bred heifers may only be purchased when the breeding cows are culled upon pregnancy testing. The purchase price of bred heifers is depicted as,

\[
\text{if (SEX = Female) and (animal = Pregnant) and (Source = Purchased) and (Pregnancy Test = Yes) then,}
\]

\[
\text{Purchase Price} = \text{(Base Monthly Price)} \times \text{(Price Adjustment For Weight)} \times \text{(Current Weight)} \times \text{PregAdjust}
\]

where,

\[
\text{PregAdjust = The price adjustment for a pregnant}
\]

186
The price paid for each open or bred heifer is summed to a variable called TotalPurchaseCost. TotalPurchaseCost represents the total annual outlay in dollars for all heifers purchased during the year.

6.6. **SellWeanlings**

The beef cow-calf operation plays an important role in the livestock economy of Virginia. Profitability in a cow-calf operation is influenced by input levels, herd genetics, and the percentage of calves that reach weaning age. When cows fail to calve or wean a calf, potential herd profitability decreases. The price received for weanling calves in Virginia depends upon the marketing method (direct, auction barns, or the Virginia Tel-O-Market), the breed of the calves, the color of the calves, the amount of shrinkage incurred, and the distance to auction barns or gathering pens. Some characteristics influencing cattle prices at auctions in other states normally do not affect the price received in Virginia since its auctions are mostly graded cattle sales (Purcell, 1997). Table 3-8 depicts the marketing adjustments received in the VBCSM for breed, color, and marketing method.

SellWeanlings adjusts the annual base selling prices per hundredweight received for 600 - 700 pound feeder steers to an average monthly price for 300 to 1000 pound steers and
heifers according to the procedures described in PurchaseLivestock by using the multipliers for weight and sex depicted in Table 3-6. After calculating the adjusted price received per hundredweight for a weanling calf, the total is divided by 100 and multiplied by the animal’s current weight. Cumulative values by sexual category (steers or heifers) are kept during the year for the total dollar amount received, the number of calves sold or transferred to the stocker herd, and the cumulative weaning weights.

Breed premiums are calculated for each purebred Angus calf or cow, while a color premium is calculated for each crossbred calf and cow if the user specified that the herd is bred for a black phenotype. These premiums are individually calculated and added to the gross income category to which the animal is assigned: steers, heifers, orphans, culled cows, stocker steers, and stocker heifers. A shrinkage allowance is then calculated for each animal according to the marketing method used. If cattle are sold at an auction barn or through the Virginia Tel-O-Market, shrinkage depends upon the distance to the closest auction barn or gathering pen and whether or not the cattle are overnighted. The amount of shrinkage or the formula used to calculate shrinkage by marketing method is depicted in Table 6-1. Auction fees for cattle sold through an auction barn or through the Virginia Tel-O-Market are depicted in Table
6-2. Annual counters are summed for the total sales and yardage fees when using either auction barn sales or the Virginia Tel-0-Market methods of marketing.

6.7. **SellOrphans**

Orphan calves are defined as calves whose dams' have died prior to weaning their calf. **SellOrphans** sells all orphan calves according to the same procedures used to sell weanling calves except that orphan calves may only be sold using the direct sales and the auction barn marketing methods.

6.8. **SellCullCows**

Price differences between different lots of culled cows are based upon differences in their expected values (Ladd & Martin, 1976). Buccola (1982) reported that price also depends upon the physical traits observed in a specific lot of cows in addition to the fundamental market forces that reflect culled cow supply and demand. The variability in prices for culled cows then reflects weight, grade, dressing percentage, health, and pregnancy in addition to sensitivity to local and regional market conditions (Mintert et al., 1990). As with other classes of cattle, producers must be familiar with the pattern of price premiums and discounts that are associated with cow traits in order to plan the optimum marketing and management strategy (Mintert et al., 1990; Nunez-Dominguez et al., 1992).
SellCullCows adjusts the annual user-specified base selling prices per hundredweight received for commercial grade cows to an average monthly price according to the procedures described in PurchaseLivestock. Prices are then adjusted for grade and pregnancy status by using the multipliers depicted in Table 6-3. Commercial grade cows are defined in this function as cows less than five years of age, while utility grade cows are defined as those five years of age or older. Cull cows may not be sold through the Virginia Tel-O-Auction. No cow marked for culling may be sold until it has finished lactating. Annual counters are kept for the sales of all cull cows, the number of cull cows sold, and the weight of cull cows sold. Shrinkage is calculated according to the methods described in SellWeanlings using the multipliers depicted in Table 6-1. If the cull cows are sold using the auction barn method, auction fees and yardage are calculated according to schedules contained in Table 6-2. Annual counters are summed for the total sales and yardage fees when using the auction barn sales method of marketing.

6.9. SellStockerCalves

The growing of stocker cattle is becoming more important in the South (Bobst et al, 1982). It is a seasonal enterprise that consists of purchasing weaned calves, putting them on pasture and supplemental feeds, and
reselling them to feedlots after time has passed. This operation can be extremely risky because feeder cattle prices can be more volatile than prices for any other class of cattle. The stocker grower's objectives should be to maximize short-term profits, create stable cash flows, and minimize the risk of loss (Russell & Dickey, 1983). The producer may express these objectives as a return on investment, profit per head, or a maximum permissible loss.

Stocker cattle are usually bought in the spring and sold in the fall since the nutritious, heavy growth of spring and summer grasses may provide more economical increases in weight (Ethridge et al., 1990). The problem with this strategy is that seasonal prices for cattle are highest in the spring (March–April) and lowest in the fall (October). It may be possible, then, for the cattle producer to increase his profits by changing his normal plans for producing and marketing stocker cattle. However, seasonal price patterns could change if enough producers altered their traditional marketing strategies for stocker calves (Ethridge et al., 1990).

SellStockerCalves adjusts the annual base selling prices per hundredweight received for 700–800 pound yearling steers to an average monthly price for 300 to 1000 pound steers and heifers according to the procedures described in PurchaseLivestock by using the multipliers for weight and sex depicted in Table 3-7. Otherwise, this
function is identical to the SellWeanlings function except that it addresses stocker calves instead of weanling calves.

6.10. **BudgetReport**

`BudgetReport` collects all information and statistics pertaining to the cow-calf operation and uses them to prepare financial reports and herd summaries at year end for year one up to year three. Since the cow-calf operation is the primary enterprise being simulated, a projected income statement showing the simulated revenues, expenses, and income or loss attributed to this enterprise is prepared. In VBCSM, the projected income statement is a planning tool that examines the profitability and the annual cash flow associated with the cow-calf operation. It can be used to determine break-even marketing prices and debt service capacity of the simulated cow-calf operation using the options specified in the dialog pages (Luening et al., 1991). Because hay is a necessary part of cow-calf production systems in Virginia, hay is included in the cow-calf projected income statement at its net value; i.e., its sales price less cost. Figure 6-1 is a projected income statement for a commercial cow-calf operation that uses 100 Angus X Hereford crossbred cows and purchases open replacement heifers whose primary breeding is Angus. Figure 6-2 shows the annual herd statistics for the operation. Herd statistics include birth statistics, culling
statistics, feed statistics, mortality statistics, production statistics, reproductive statistics, body condition scores for the herd at the start of the last breeding season, and the age break-down of the herd.

BudgetReport collects and uses statistics pertaining to the stocker calf operation to prepare partial budget reports and stocker calf summaries at year-end for up to three years. The profitability of the stocker calf operation and its financial impact upon debt repayment should be tested before actually adding it to the farm business. Partial budgets are a simple but accurate and powerful tool for testing these impacts (Luening et al., 1991). They consider only those costs and returns that change as a result of the proposed plan. Figure 6-3 is the projected partial budget for the stocker calf operation that is to be added to the farm business. Figure 6-4 summarizes its annual statistics and includes the number of stockers on hand, feed statistics, mortality statistics, production statistics, and breakeven analysis. BudgetReport also includes a sensitivity or “IF-THEN” analysis that shows the effects upon profit (loss) of changes in the prices paid or received per hundredweight for the stocker calves.

6.11. **ResetMarketVariables**

On the 365\textsuperscript{th} day of each simulation year, ResetMarketVariables resets to zero the variables and counters in the
marketing module that are used to collect annual statistics.
Literature Cited


Virginia Cattlemen’s Association. 1998. 1998 Feeder Cattle Tel-O-Auction Field Sales. Daleville, VA.


<table>
<thead>
<tr>
<th>Farm:</th>
<th>BlueBerry Ridge Farms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address:</td>
<td>Montgomery County, Virginia</td>
</tr>
<tr>
<td>Farm Owner:</td>
<td>Mr. W. D. Doe</td>
</tr>
<tr>
<td>Farm Manager:</td>
<td>Ms. Shelly Doe</td>
</tr>
<tr>
<td>Livestock Manager:</td>
<td>Ms. Catherine Doe</td>
</tr>
<tr>
<td>Farm Size:</td>
<td>300 Acres</td>
</tr>
<tr>
<td>Max Herd Size w/ Replacements</td>
<td>100</td>
</tr>
<tr>
<td>Calving Season:</td>
<td>Spring</td>
</tr>
</tbody>
</table>

Year End Statements for Year: 1

PROJECTED INCOME/LOSS STATEMENT FOR THE COW-CALF OPERATION

<table>
<thead>
<tr>
<th>Projected Income:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Steer Revenue..........................12487.26</td>
</tr>
<tr>
<td>Total Heifer Revenue..........................11662.37</td>
</tr>
<tr>
<td>Total Orphan Calf Revenue.................... 195.57</td>
</tr>
<tr>
<td>Total Culled Cow Revenue...................... 5493.49</td>
</tr>
<tr>
<td>Total Projected Gross Revenue................30899.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Projected Expenses:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Livestock Purchases..................5500.30</td>
</tr>
<tr>
<td>Pasture Rental........................5900.00</td>
</tr>
<tr>
<td>Pasture Maintenance...................3826.15</td>
</tr>
<tr>
<td>Supplements (at net)........................1805.10</td>
</tr>
<tr>
<td>Salt and Minerals..........................544.00</td>
</tr>
<tr>
<td>Implants..................................61.20</td>
</tr>
<tr>
<td>Veterinary Service &amp; Medicine...........1225.00</td>
</tr>
<tr>
<td>Livestock Supplies.......................200.00</td>
</tr>
<tr>
<td>Breeding Bull Expenses..................960.00</td>
</tr>
<tr>
<td>Marketing Transportation Charges:</td>
</tr>
<tr>
<td>Calves....................................328.50</td>
</tr>
<tr>
<td>Culled Cows..................................67.60</td>
</tr>
<tr>
<td>Marketing Expenses:</td>
</tr>
<tr>
<td>Calves:</td>
</tr>
<tr>
<td>Commissions &amp; Fees.................556.50</td>
</tr>
<tr>
<td>Yardage...................................714.32</td>
</tr>
<tr>
<td>Culled Cows:</td>
</tr>
<tr>
<td>Commissions &amp; Fees ...................65.00</td>
</tr>
<tr>
<td>Yardage..................................172.03</td>
</tr>
<tr>
<td>Building &amp; Fence Repairs...................250.00</td>
</tr>
<tr>
<td>Utilities..................................125.00</td>
</tr>
<tr>
<td>Labor......................................6400.00</td>
</tr>
<tr>
<td>Payroll Taxes................................489.60</td>
</tr>
<tr>
<td>Machinery Expenses..........................1000.00</td>
</tr>
<tr>
<td>Total Operating Expenses................30190.30</td>
</tr>
<tr>
<td>Interest Expenses..................................1930.19</td>
</tr>
<tr>
<td>Total Projected Expenses................32120.49</td>
</tr>
<tr>
<td>Net Loss...................................-1220.50</td>
</tr>
</tbody>
</table>

Figure 6-1. VBCSM Projected Cow-Calf Income Statement.
Statistics of the Cow-Calf Operation

ANNUAL BIRTH STATISTICS:

Annual Number of Calves Born: 79  
Average Birthweight of Steers : 96  
Average Birthweight of Heifers: 87  
Average Birthweight of All Calves: 91

ANNUAL CULLING STATISTICS:

Number of Culled Cows Sold: 13  
Cull Cows on Hand: 0  
Total Weight of Culled Cows Sold: 14953 Lbs  
Average Weight of Culled Cows Sold: 1150 Lbs  
Culling Rate: 13.00 %

ANNUAL FEED STATISTICS:

Estimated Forage Available for Grazing: 544.60 Tons  
Forage Consumed, Cow-Calf Ops: 419.14 Tons  
Hay Grown on the Farm: 154.52 Tons  
Hay Consumption: 68.72 Tons  
Concentrates Consumed: 0.85 Cwt.

ANNUAL MORTALITY STATISTICS:

Number of Dead Heifers: 1  
Number of Dead Steers: 4  
Number of Dead Calves: 5  
Death Losses of Breeding Females: 3  
Death Rate, Cows: 3.00 %  
Death Rate, Calves: 6.33 %

ANNUAL PRODUCTION STATISTICS:

Number of Orphan Calves Sold: 1  
Number of Steer Calves Sold: 34  
Number of Heifer Calves Sold: 39  
Total Number of Weanling Calves Sold: 73  
Total Weight of Steer Calves: 16901 LBS  
Total Weight of Heifer Calves: 18474 LBS  
Total Weight of All Calves: 35376 LBS  
Average Weight of Steer Calves: 497 LBS  
Average Weight of Heifer Calves: 473 LBS  
Average Weight of Weanling Calves: 484 LBS  
Average Daily Gain, Spring Steers: 2.07 LBS  
Average Daily Gain, Spring Heifers: 1.99 LBS  
Average Daily Gain, Spring Calves: 2.03 LBS  
Average Weaning Age, Spring Calves: 193 Days

Figure 6-2. VBCSM Annual Projected Cow-Calf Statistics.
ANNUAL REPRODUCTION STATISTICS:

Length of the Breeding Season for Spring Calving: 90 Days

At the End of the Last Breeding Season:

<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Cycling Breeding Cows</td>
<td>97</td>
</tr>
<tr>
<td>Number of Cycling Breeding Cows Open</td>
<td>13</td>
</tr>
<tr>
<td>Number of Replacement Cows</td>
<td>0</td>
</tr>
</tbody>
</table>

At Current Year-End:

<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulls on Hand</td>
<td>3</td>
</tr>
<tr>
<td>Breeding Cows on Hand, All Ages</td>
<td>100</td>
</tr>
<tr>
<td>Pregnant Cows on Hand</td>
<td>84</td>
</tr>
<tr>
<td>Number More than 45 Days Pregnant, Aborting</td>
<td>0</td>
</tr>
<tr>
<td>Number of Cows Failing to Calve</td>
<td>3</td>
</tr>
<tr>
<td>Number of Cows Failing to Wean a Calf</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Animals Experiencing Dystocia</td>
<td></td>
</tr>
<tr>
<td>2 Year Olds</td>
<td>4</td>
</tr>
<tr>
<td>3 Year Olds</td>
<td>1</td>
</tr>
<tr>
<td>4 Year Olds</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Bred Heifers Purchased</td>
<td>0</td>
</tr>
<tr>
<td>Replacement Nonpregnant Heifers Purchased</td>
<td>16</td>
</tr>
<tr>
<td>Number of Within Herd Replacements Selected</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Length of Gestation</td>
<td>286.34</td>
</tr>
<tr>
<td>Average No. Services/Conception</td>
<td>1.63</td>
</tr>
<tr>
<td>Pregnancy Rate</td>
<td>88.66 %</td>
</tr>
<tr>
<td>Calving Rate</td>
<td>79.00 %</td>
</tr>
<tr>
<td>Weaning Rate</td>
<td>73.00 %</td>
</tr>
<tr>
<td>Average Calving Date, Spring Calving</td>
<td></td>
</tr>
<tr>
<td>Day of The Year</td>
<td>DAY 117</td>
</tr>
<tr>
<td>Days Since the Start of the Breeding Season</td>
<td></td>
</tr>
<tr>
<td>Heifer Breeding Season</td>
<td>340 Days</td>
</tr>
<tr>
<td>Regular Breeding Season</td>
<td>315 Days</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Heifers Eligible To Breed at Start of Breeding Season</td>
<td>16</td>
</tr>
</tbody>
</table>

Heifers Cycling During Heifer Breeding Season, Spring Calving:

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>During the First Third</td>
<td>15</td>
</tr>
<tr>
<td>During the Second Third</td>
<td>15</td>
</tr>
<tr>
<td>During the Last Third</td>
<td>15</td>
</tr>
</tbody>
</table>

Heifers Pregnant During Heifer Breeding Season, Spring Calving:

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>During the First Third</td>
<td>8</td>
</tr>
<tr>
<td>During the Second Third</td>
<td>12</td>
</tr>
<tr>
<td>During the Last Third</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Cows</td>
<td>81</td>
</tr>
</tbody>
</table>

Cows Cycling During Regular Breeding Season, Spring Calving:

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>During the First Third</td>
<td>70</td>
</tr>
<tr>
<td>During the Second Third</td>
<td>78</td>
</tr>
<tr>
<td>During the Last Third</td>
<td>80</td>
</tr>
</tbody>
</table>

Figure 6-2 (Cont.) VBCSM Annual Projected Cow-Cow Statistics.
Statistics of the Cow-Calf Operation (Cont.)

ANNUAL REPRODUCTION STATISTICS (Cont.):

Cows Pregnant During Regular Breeding Season, Spring Calving:
During the First Third: 41
During the Second Third: 59
During the Last Third: 70

Number of Heifers and Cows Estimated to be Chronically Unable to Maintain a Pregnancy: 2

BCS Of Nonpregnant Breeding COWS & HEIFERS At THE End OF THE LAST BREEDING SEASON:

<table>
<thead>
<tr>
<th>BCS</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

AVERAGE AGE OF THE MATURE BREEDING HERD: 2.71 Years

AGE DISTRIBUTION OF THE BREEDING HERD:

<table>
<thead>
<tr>
<th>Age Group</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breeding Cows on Hand, Less than 1 Year</td>
<td>15</td>
</tr>
<tr>
<td>Breeding Cows on Hand, Yearlings</td>
<td>15</td>
</tr>
<tr>
<td>Breeding Cows on Hand, 2 Year Olds</td>
<td>29</td>
</tr>
<tr>
<td>Breeding Cows on Hand, 3 Year Olds</td>
<td>22</td>
</tr>
<tr>
<td>Breeding Cows on Hand, 4 Year Olds</td>
<td>10</td>
</tr>
<tr>
<td>Breeding Cows on Hand, 5 Year Olds</td>
<td>6</td>
</tr>
<tr>
<td>Breeding Cows on Hand, 6 Year Olds</td>
<td>2</td>
</tr>
<tr>
<td>Breeding Cows on Hand, 9 Year Olds</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 6-2 (Cont.) VBCSM Annual Projected Cow-Calf Statistics.
**PROJECTED PARTIAL BUDGET**
**FOR THE STOCKER CALF OPERATION**

Additional Projected Income (Loss):

<table>
<thead>
<tr>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Stocker Steer Revenue</td>
<td>22875.70</td>
</tr>
<tr>
<td>Total Stocker Heifer Revenue</td>
<td>11530.14</td>
</tr>
<tr>
<td>Total Additional Projected Revenue</td>
<td>35828.93</td>
</tr>
</tbody>
</table>

Additional Projected Expenses:

<table>
<thead>
<tr>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost of Stocker Steers Sold</td>
<td>15912.82</td>
</tr>
<tr>
<td>Cost of Stocker Heifers Sold</td>
<td>8282.74</td>
</tr>
<tr>
<td>Salt and Minerals</td>
<td>88.65</td>
</tr>
<tr>
<td>Implants</td>
<td>203.40</td>
</tr>
<tr>
<td>Veterinary Service &amp; Medicine</td>
<td>399.73</td>
</tr>
<tr>
<td>Livestock Supplies</td>
<td>213.00</td>
</tr>
<tr>
<td>Market Transportation Charges</td>
<td>319.50</td>
</tr>
<tr>
<td>Marketing Expenses:</td>
<td></td>
</tr>
<tr>
<td>Commissions &amp; Fees</td>
<td>697.50</td>
</tr>
<tr>
<td>Yardage</td>
<td>1031.96</td>
</tr>
<tr>
<td>Building &amp; Fence Repairs</td>
<td>213.00</td>
</tr>
<tr>
<td>Utilities</td>
<td>124.25</td>
</tr>
<tr>
<td>Labor</td>
<td>1988.00</td>
</tr>
<tr>
<td>Payroll Taxes</td>
<td>152.08</td>
</tr>
<tr>
<td>Insurance Expenses</td>
<td>90.50</td>
</tr>
<tr>
<td>Machinery Expenses</td>
<td>710.00</td>
</tr>
<tr>
<td>Additional Operating Expenses</td>
<td>30336.63</td>
</tr>
</tbody>
</table>

Interest Expenses                    | 2062.89

Total Additional Expenses            | 32399.52

Net Additional Revenue               | 3429.41

---

**Figure 6-3.** VBCSM Projected Partial Budget For the Stocker Calf Operation.
## Stocker (Retained) Calf Statistics

Current number of Stocker Calves on Hand: 73

### ANNUAL FEED STATISTICS:

- Estimated Forage Available for Grazing: 125.46 Tons
- Forage Consumed, Stocker Ops: 29.44 Tons
- Hay Consumed, Stocker Ops: 26.75 Tons
- Concentrates Consumed, Stocker Ops: 178.31 Cwt.

### ANNUAL MORTALITY STATISTICS:

Number of Dead Stocker Calves: 0
Number of Dead Stocker Heifers: 0
Number of Dead Stocker Steers: 0

### ANNUAL PRODUCTION & MARKETING STATISTICS:

- Number of Stocker Calves Sold: 71
- Number of Stocker Steers Sold: 42
- Number of Stocker Heifers Sold: 29
- Average Weight of Stocker Calves Sold: 668 LBS
- Average Weight of Stocker Steers Sold: 709 LBS
- Average Weight of Stocker Heifers Sold: 608 LBS
- Average Daily Gain of Stocker Calves Sold: 1.60 LBS
- Average Daily Gain of Stocker Steers: 1.63 LBS
- Average Daily Gain of Stocker Heifers: 1.55 LBS
- Spring Born Stocker Calves Are Sold On: DAY 64
- Average COST of Stocker Calves Sold Includes Death Losses (if any), $/Cwt Sold: 51.01
- Average Price RECEIVED For Stocker Calves After Applying Premiums/Discounts, $/Cwt Sold: 72.53
- Average Stocker Calf Price REQUIRED to BREAK-EVEN, $/Cwt sold: 68.30

### Figure 6-4.

VBCSM Annual Projected Stocker Calf Statistics.
Table 6-1. The Amount Of Shrinkage For Cattle By The Marketing Method Used In The Virginia Beef Cattle Simulation Model (Self & Gay, 1972).

<table>
<thead>
<tr>
<th>METHOD</th>
<th>SHRINKAGE MULTIPLIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>0.03</td>
</tr>
<tr>
<td>Auction Barn - Distance Less than 50 Miles/No Overnighting</td>
<td>0.022</td>
</tr>
<tr>
<td>Auction Barn - Distance Less than 50 Miles/Overnighting</td>
<td>0.042</td>
</tr>
<tr>
<td>Auction Barn - Distance 50 Miles or More/No Overnighting</td>
<td>0.022 + (0.0038 * (Distance/50))</td>
</tr>
<tr>
<td>Auction Barn - Distance 50 Miles or More/Overnighting</td>
<td>0.042 + (0.0038 * (Distance/50))</td>
</tr>
<tr>
<td>Virginia Tel-O-Market Gathering Pen Distance Less than 50 Miles/No Overnighting</td>
<td>0.022</td>
</tr>
<tr>
<td>Virginia Tel-O-Market Gathering Pen Distance Less than 50 Miles/Overnighting</td>
<td>0.042</td>
</tr>
<tr>
<td>Virginia Tel-O-Market Gathering Pen Distance 50 Miles or More/No Overnighting</td>
<td>0.022 + (0.0038 * (Distance/50))</td>
</tr>
<tr>
<td>Virginia Tel-O-Market Gathering Pen Distance 50 Miles or More/Overnighting</td>
<td>0.042 + (0.0038 * (Distance/50))</td>
</tr>
</tbody>
</table>
Table 6-2. Auction and Virginia Tel-O-Market Fees used in the Virginia Beef Cattle Simulation Model (McKinnon, 1998; Virginia Cattlemen’s Association, 1998).

<table>
<thead>
<tr>
<th>FEE ITEM</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Commission</td>
<td>$ 3.00</td>
</tr>
<tr>
<td>Check-Off Fee</td>
<td>$ 1.00</td>
</tr>
<tr>
<td>Fee Adjustment for Calves over 600 lbs.</td>
<td>$ 1.00</td>
</tr>
<tr>
<td>Grading Fee (Auction Barn)</td>
<td>$ 0.50</td>
</tr>
<tr>
<td>Yardage</td>
<td>3% of total receipts</td>
</tr>
<tr>
<td>Virginia Tel-O-Market Sales Commissions</td>
<td>$ 1.00</td>
</tr>
<tr>
<td>(Virginia Cattlemen’s Association)</td>
<td></td>
</tr>
<tr>
<td>Virginia Tel-O-Market Telephone Charges</td>
<td>$ 1.50</td>
</tr>
<tr>
<td>Grading Fee (Virginia Tel-O-Market)</td>
<td>$ 0.75</td>
</tr>
<tr>
<td>Local Association Fees</td>
<td>$ 0.25</td>
</tr>
</tbody>
</table>
Table 6-3. Multiplicative Price Adjustments Used To Calculate The Price Received For Culled Cows In The Virginia Beef Cattle Simulation Model (FAPRI, 1997; Commodity Research Bureau, 1997).

<table>
<thead>
<tr>
<th>Livestock Class</th>
<th>Multiplicative Price Adjustments</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMERCIAL GRADE COWS</td>
<td>1.000</td>
</tr>
<tr>
<td>UTILITY GRADE COWS</td>
<td>0.891</td>
</tr>
<tr>
<td>PREGNANT COWS</td>
<td>1.910</td>
</tr>
</tbody>
</table>
CHAPTER 7. Testing The Virginia Beef Cattle Simulation Model

7.1. Introduction

7.2. Materials & Methods
   7.2.1. Herd & Management Inputs
   7.2.2. Forage & Nutritional Inputs
   7.2.3. Economic Inputs
   7.2.4. Mathematical Model

7.3. Results & Discussion
   7.3.1. Analysis of Variance
      7.3.1.1. Calving Season Effects
      7.3.1.2. Length of Breeding Season Effects
      7.3.1.3. Culling Policy Effects
      7.3.1.4. Calving Season x Length of Breeding Season Effects
      7.3.1.5. Calving Season x Culling Policy Effects
      7.3.1.6. Length of Breeding Season x Culling Policy Effects
      7.3.1.7. Year Effects
7.3.2. Descriptive Statistics of Biological Traits

7.4. Conclusion

7.4.1. Potential Source Code Weaknesses

7.5. Implications

Literature Cited
Chapter 7

Testing The Virginia

Beef Cattle Simulation Model

Abstract: A modified factorial model with two calving seasons, three lengths of breeding season, four culling policies, and a year effect was used to examine the results of test simulations of the Virginia Beef Cattle Simulation Model. Calving season, length of breeding season, and culling policy influenced weanling steer and heifer revenue, cow-calf net income, stocker steer and heifer revenue, stocker net income, and total net income. Length of breeding season influenced culling rate, pregnancy rate, weaning rate, weaning weight, average weaning age of the calf, and the number of stocker calves sold. Culling policy influenced culling rate, pregnancy rate, weaning rate, the number of stocker calves sold, and the number of replacement heifers. Two-way interactions had small influences upon some income traits but there were no important three-way interactions influencing any trait in VBCSM. Year effects were only important for culled cow revenue, cow-calf net income, stocker net income, and culling rate. Several descriptive statistics suggested that the program code was acted in a consistent manner across season. Several weaknesses in the program were discovered. Fall calving appeared to be more profitable than spring calving while retaining spring calves appeared to be more profitable than retaining fall calves.

7.1. Introduction

The objectives of this chapter are to describe methods used to test the Virginia Beef Cattle Simulation Model, to present results from the testing, and to summarize economical and biological implications of such testing. The first step in accomplishing these objectives was to select base values for biological and economic inputs that would be
realistic for a cow-calf producer in the Mountain Region of Virginia. The next was to choose three management variables (two calving seasons, three breeding season lengths, and four culling policies) to simulate in all possible combinations in order to determine their effects and interactions on biological and economic traits. Data then were collected for three sequential years from 10 replicates of each calving season x breeding season length x culling policy combination and subjected to analysis of variance. Descriptive statistics were examined to determine whether the VBCSM worked in a consistent fashion across calving seasons, lengths of breeding season, and culling policies. Finally, potential production strategies were evaluated for profitability and efficiency. The results of this statistical analysis and their implications are presented and discussed in this chapter.

7.2. Materials and Methods

Animal production and enterprise profit or loss were simulated for three management practices (two calving seasons, three breeding season lengths, and four culling policies), with 10 independent runs of three consecutive years’ duration for each of the 24 management practice combinations. All other management, biological, and economic factors were held constant at input values as described in the following sections of 7.2. Simulated
biological and financial data were then subjected to analysis of variance to determine calving season, breeding season length, and culling policy effects and interactions.

7.2.1. **Herd & Management Inputs**

The foundation herd consisted of 100 F₁ crossbred Angus x Hereford cows including replacement heifers. Gelbvieh bulls were mated to mature breeding cows while Angus bulls were mated to heifers. Breeding was by natural service. All replacement heifers were selected from within the herd from available pre-weanling heifer calves sired by both Gelbvieh and Angus bulls. In addition to four alternative policies under which the cows were culled according to their ability to become pregnant and wean a calf, cows were automatically culled upon reaching 15 years of age. However, no lactating cow was culled until after its calf was weaned. In any one simulation, there was only one annual calving season, but spring calving and fall calving systems were simulated in separate runs. The first day of the breeding season for spring calving was June 16, while the first day of breeding for fall calving was December 1. Spring calves were weaned on November 5 while fall calves were weaned on June 2. The breeding season for heifers started and ended 25 days earlier than the breeding season for older females. This first day of the breeding season for heifers was May 22 for spring calving and November 5 for
fall calving.

Four bulls were available to service the breeding herd. Each bull was used for three years and had a salvage value of 60% of its original $1200 cost. Since black colored animals have received a marketing premium in recent years, all calf generations were bred for a black phenotype. After weaning, calves were retained for 120 days in a pasture-based stocker calf program. Anabolic implants were given to pre-weanling steer calves, stocker steers, and stocker heifers.

7.2.2. Forage and Nutritional Inputs

The simulated farm whose owner-operator is 100% responsible for its maintenance was located in the Mountain Region of Virginia. It consisted of 300 acres divided between 220 acres of permanent pasture, 75 acres of seasonally grazable hay lands, and five acres of farmstead and woodlands. Farm erosion was light and pastures were fertilized and maintained in an average manner. Although the terrain was hilly, pasture slope was generally less than 10 degrees. The pastures were continuously grazed by the breeding herd, and the amount of weed infestation within the pastures and hay lands was considered to be light. Cattle were not allowed to graze the hay lands between March 1st and July 1st annually. The pasture forage contained 45% fescue/ladino clover, 40% orchardgrass/ladino clover, and
15% kentucky bluegrass/white clover. When necessary, pre-weanling and stocker calves received a supplemental feed consisting of 50% Fescue Hay, 25% Orchardgrass Hay, and 25% Broiler Litter. Cows and weaned replacements were supplemented only with hay. Hay was assumed to cost $31.03/ton to produce and had a sales value of $60.00/ton. Annual rental rates for pasture were $20.00/acre.

7.2.3. Economic Inputs

Culled cows and orphan calves were sold in auction barn sales while weanling calves and stocker calves were sold through the Virginia Tel-O-Market. It was assumed that the simulated farm was located 40 miles from the auction barn or the Virginia Tel-O-Market gathering pens and that cattle were not over-nighted but instead sold the same day that they were delivered. Neither the weanling calves nor the stocker calves were hedged to protect the prices received for them. The base annual sales prices per hundredweight received for the various classes of simulated cattle are:

<table>
<thead>
<tr>
<th>YEAR</th>
<th>600 – 700 LB. STEER CALVES ($/CWT.)</th>
<th>700 – 800 LB. STOCKER STEERS ($/CWT.)</th>
<th>COMMERCIAL GRADE COWS ($/CWT.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>72.50</td>
<td>70.00</td>
<td>36.00</td>
</tr>
<tr>
<td>2</td>
<td>70.00</td>
<td>68.00</td>
<td>40.00</td>
</tr>
<tr>
<td>3</td>
<td>71.00</td>
<td>69.00</td>
<td>41.00</td>
</tr>
</tbody>
</table>

Price differentials for different weights and sexes of cattle were defined in a previous chapter. Table 3-6 reported these values for weaned calves while Table 3-7
reported these values for stocker calves. Monthly price differentials were presented in Table 3-8. Price adjustments for breed and color premiums as well as for culled and replacement cows were shown in Table 3-9. Culled and replacement cow prices were also adjusted according to Table 6-3.

Default values in VBCSM were used for the costs of production for the cow-calf operation. These costs are:

<table>
<thead>
<tr>
<th>ITEM</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building &amp; Fence Repairs</td>
<td>$ 7.00/Cow</td>
</tr>
<tr>
<td>Federal Payroll Tax Rate</td>
<td>7.65%</td>
</tr>
<tr>
<td>Anabolic Implants</td>
<td>$ 1.25/Each</td>
</tr>
<tr>
<td>Livestock Supplies</td>
<td>$ 2.00/Cow</td>
</tr>
<tr>
<td>Non-Crop Machinery Costs</td>
<td>$ 10.00/Cow</td>
</tr>
<tr>
<td>Pasture Maintenance</td>
<td>$ 12.97/Acre</td>
</tr>
<tr>
<td>Salt &amp; Minerals</td>
<td>$ 8.00/Cwt.</td>
</tr>
<tr>
<td>Culled Cow Market Transportation</td>
<td>$ 5.20/Cow</td>
</tr>
<tr>
<td>Market Calf Transportation</td>
<td>$ 4.50/Calf</td>
</tr>
<tr>
<td>Utilities</td>
<td>$ 1.25/Cow</td>
</tr>
<tr>
<td>Labor Cost</td>
<td>$ 8.00/ Hour</td>
</tr>
<tr>
<td>Labor Hours</td>
<td>5 Hours/Cow</td>
</tr>
<tr>
<td>Veterinary Services &amp; Medicine</td>
<td>$ 12.25/Cow</td>
</tr>
<tr>
<td>Bank Interest Rates</td>
<td>9.50%</td>
</tr>
</tbody>
</table>

To compute interest expenses, it was assumed that the size of the operating loan equaled 80% of estimated annual operating expenses.

Default values in VBCSM were also used for the costs of producing stocker calves. These values are:

<table>
<thead>
<tr>
<th>ITEM</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building &amp; Fence Repairs</td>
<td>$ 3.00/Stocker</td>
</tr>
<tr>
<td>Livestock Supplies</td>
<td>$ 3.00/Stocker</td>
</tr>
<tr>
<td>Non-Crop Machinery Costs</td>
<td>$ 10.00/Stocker</td>
</tr>
<tr>
<td>Maintenance Costs for Rented Pasture</td>
<td>$ 12.97/Acre</td>
</tr>
</tbody>
</table>
To compute interest expenses, it was assumed that the size of the operating loan equaled 80% of the estimated operating expenses incurred during the time that stocker calves were being grown on the farm.

7.2.4. Mathematical Model

The effects of calving season, length of the breeding season, culling policy, and year on income and production variables were examined by means of F-tests from analysis of variance using a modified factorial design (GLM Procedure, SAS, Littell et al., 1990). The model may be symbolized as:

\[ Y_{ijklm} = S_i + L_j + C_k + SL_{ij} + SC_{ik} + LC_{jk} + SLC_{ijk} + Yr_l + e_{ijklm} \]

where,

\( S_i \) = Calving season (spring or fall)

\( L_j \) = Length of the breeding season in days (45-60-90 days)

\( C_k \) = The culling policy being tested

0 = Open On Pregnancy Testing

1 = Failed To Wean A Calf In Any 1 Year

2 = Failed To Wean A Calf In Any 2 Years

3 = Failed To Wean A Calf In Any 3 Years

\( SL_{ij} \) = The interaction between calving season (spring or fall) and length of the breeding
season in days (45-60-90 days)

SC\textsubscript{ik} = The interaction between calving season (spring or fall) and culling policy (0, 1, 2, or 3)

LC\textsubscript{jk} = The interaction between length of breeding season in days (45-60-90 days) and culling policy (0, 1, 2, or 3)

SLC\textsubscript{ijk} = The interaction between calving season (spring or fall), length of breeding season in days (45-60-90 days), and culling policy (0, 1, 2, or 3)

Yr\textsubscript{l} = Year within replicate within SLC class

e\textsubscript{ijklm} = The error term; i.e., the difference between predicted values and actual values

Each of the 24 possible combinations of S\textsubscript{i} x L\textsubscript{j} x C\textsubscript{k} was simulated 10 times with results captured for three consecutive years in each run. Year effects were analyzed using a time series analysis to observe any trends from the annual changes in cattle prices.

7.3. Results and Discussion

7.3.1. Analysis of Variance

7.3.1.1. Calving Season Effects

Spring versus fall calving systems differed importantly for weanling steer revenue, weanling heifer revenue, cow-calf net income, stocker steer revenue, stocker heifer revenue.
revenue, stocker net income, and total net income (Table 7-1). Calving season had no important influences on culled cow revenue, birth weights, cow death rate, calf death rate, pregnancy rate, weaning rate, or dystocia rate.

As shown in Table 3-8, cattle prices varied from month to month when indexed on an annual value. Fall calves weaned and sold in June received approximately a seven percent higher price per pound than spring calves weaned and sold in November. Spring calves that were retained 120 days in a stocker calf program sold for approximately nine percent more per pound than fall calves retained for 120 days and sold in October. These price differentials were partially responsible for the difference in profitability between fall calves and spring calves at weaning and when sold as stocker calves.

Fall calves were significantly heavier than spring calves at weaning and at their final stocker sales weight (Table 7-1). The weight differences between fall and spring calves resulted from the fall calves being on average 40 days older at weaning than the spring calves. At their final sales weight, the fall calved stockers averaged 783 pounds while the spring calved stockers averaged 675 pounds.

Cow-calf hay consumption and the amount of concentrates fed to the calves were much greater in fall calving than in spring calving systems (Table 7-1). Calves and cows suckling calves required more supplemental feeds to meet
their nutritional requirements for body maintenance and growth during the cold winter months when forage growth was at a minimum than they did during the spring and early summer months when forage growth rates peaked. Since the end result was that less pasture forage was available for fall calves and their dams than was available for spring calves and their dams, more supplemental feeds were required during the pre-weaning period for fall calves. In the stocker category, however, the spring calved stockers consumed a higher proportion of supplemental feeds than fall calved stockers. In this case, the spring calved stockers were maintained over the winter months while the fall calved stockers were not.

As shown in Table 7-1, there were on average more cows that were eligible to breed (13 months of age and older) during the breeding season for fall calving than for spring calving. When selecting replacement heifers from within the herd, VBCSM allowed selections to be made only from available pre-weanling heifers. In the assumptions of the test simulations, fall calved heifers were on average 40 days older than spring calves. This meant that in each year, there were 40 extra days under the fall calving program that replacement heifers could be selected than in spring calving programs. Consequently, in the event of late term abortions and/or a cow’s death in fall calving herds, there was less likelihood of being required to wait for the
ensuing year’s calf crop before selecting needed replacements. This, in turn, translated into more cows being mature enough to breed during a fall calving season in VBCSM than during a spring calving season.

7.3.1.2. Length of Breeding Season Effects

Length of breeding season effects, as shown in Table 7-2, were important for weanling steer revenue, weanling heifer revenue, cow-calf net income, stocker steer revenue, stocker heifer revenue, stocker net income, and total net income. Length of breeding season effects were also important for culling rate, pregnancy rate, weaning rate, weaning weight, the average weaning age of the calf, and the number of stocker calves sold (Table 7-2). Length of breeding season effects did not appear to be important for cull cow income, birth weight, cow death rates, calf death rates, cow-calf hay consumption, and calf concentrate consumption. Nor did length of breeding season effects appear to be important in relation to the number of cows that were eligible to breed and the number of cows that aborted or failed to wean a calf. They also did not appear to influence dystocia or the average age of mature breeding cows within the herd.

The probable reason that each income category was affected by the length of the breeding season was the increase in the number of cows that become pregnant as the
breeding season increased from 45 days to 90 days (Table 7-2). As pregnancy rate increased, weaning rate increased as well but culling rate decreased. With increased weaning rate, more stocker calves became available for sale because more calves were available at weaning but fewer replacement heifers were required.

7.3.1.3. Culling Policy Effects

There are four culling policies available for user selection in VBCSM: 0 = Open On Pregnancy Testing; 1 = Fails To Wean A Calf In Any 1 Year; 2 = Fails To Wean A Calf In Any 2 Years; 3 = Fails To Wean A Calf In Any 3 Years.

In this test of VBCSM, culling policy had an important effect upon weanling steer revenue, weanling heifer revenue, culled cow revenue, cow-calf net income, stocker steer revenue, stocker heifer revenue, stocker net income, and total net income as shown in Table 7-3. The effect of culling policy was also important for culling rate, pregnancy rate, weaning rate, the number of replacement heifers selected, and the total number of stocker calves that were sold (Table 7-3).

The probable reason for the culling policy effect upon these traits is that as culling criteria are relaxed, more and more nonpregnant cows remain in the herd causing the pregnancy rate to decrease. Because fewer animals are pregnant, fewer calves are born and, correspondingly, the
number of weaned calves must also decrease. This is borne out by the decrease in the weaning rate associated with the more liberal policies. Additionally, as the culling rate becomes more lax, fewer replacement heifers are required resulting in more weaned calves available for transfer to the stocker herd. This is confirmed by the increase in the number of stocker calves that are available for sale and the decrease in the number of replacement heifers required for herd maintenance in the more liberal culling policies (Table 7-3).

However, this also indicates a potential weakness in the program code. As currently programmed, the record structure variable FAIL is incremented on December 31\textsuperscript{st} for each nonpregnant cow that was eligible to breed during the simulation year, even if she weaned a calf during the year.

In culling policies 1, 2, and 3, the record structure variable FWEAN is also incremented for each cow whenever she fails to wean a calf for any reason. Cows are then marked for culling on December 31 whenever the total of their record structure variables FAIL plus FWEAN equals 1, 2, or 3 for culling policies 1, 2, or 3, respectively. Under this logic, the possibility exists for both record structure variables FAIL and FWEAN to be incremented for each cow during the current simulation year while under actual conditions, the cattle producer may not be able to tell if a cow was pregnant on December 31\textsuperscript{st}. Perhaps, then, a closer
approximation of reality would be to not cull cows that
failed to become pregnant or to maintain a pregnancy in any
year until the following year at some predetermined date
prior to the start of the breeding season.

7.3.1.4. Calving Season x Length of Breeding
Season Effects

The calving season x length of breeding season
interaction effect was significant but not large for steer
revenue, heifer revenue, cow-calf net income, stocker net
income, and total net income, as shown in Figure 7-1a and
Figure 7-1b (a, b, c). An increase in length of the
breeding season generally increased gross income from weaned
steers and heifers. In only one case, increasing breeding
season length from 60 to 90 days in spring calving, did
steer revenue decrease for which there is no apparent
explanation. For cow-calf net income, lengthening the
breeding season had a larger effect in fall calving than in
spring calving, whereas the reverse was true for stocker net
income. Again, for total net income, lengthening the
breeding had a larger effect in fall calving than in the
spring. However, the magnitude of the effect between the
two calving seasons was greatly reduced.

Overall, however, these significantly affected income
traits are likely influenced by the interaction between
changes in pregnancy rate as length of breeding season
increases (Table 7-2) and the differences in prices received for fall calves at weaning compared to spring calves and spring calved stockers compared to fall calved stockers. As pregnancy rates increase, so do weaning rates. This results in fewer cows being culled and more weanlings available for the stocker calf program. Because the interaction affects both enterprise net incomes, it also exerts an effect upon total net income.

7.3.1.5. Calving Season x Culling Policy Effects

As shown in Figure 7-2a, b, c, and d, calving season x culling policy interactions were significant for weanling heifer revenue, cow-calf net income, stocker heifer revenue, and stocker net income. Higher prices for fall calved than spring calved heifers served to magnify culling system effects on heifer revenue. The reason for the shift in ranks of culling policies between calving seasons is not known. Lower prices for fall calved than spring calved stockers, however, served to deflate culling system effects on stocker heifer revenue. Again, price differences are the likely cause of inflation in the changes between culling systems for cow-calf net income and the deflation in the changes between culling systems for stocker net income between spring and fall calving strategies. These impacts were probably enhanced by increased numbers of weanling heifers being transferred to the stocker herd as the culling
policy became more lax.

7.3.1.6. Length of Breeding Season x Culling Policy Effects

The interaction between length of breeding season and culling policy was significant for cow-calf net income and culling rate. As shown in Figures 7-3a and 7-3b, the cow-calf net income increased as the culling policy became more lax and as the length of the breeding season increased. This interaction for cow-calf net income is attributable to the difference in length of breeding season effect upon culling policies zero and three compared to its effect in culling policies one and two. As breeding season lengthened, culling rate decreased for culling policies 1 and 0 but changed very little for the more lax culling polices, 2 and 3. In part for cow-calf net income, the interaction is due to larger differences among culling policies with 90 day breeding seasons compared to a 45 or a 60 day breeding seasons. Although the magnitude of differences among culling policies differed according to the length of breeding season, they did not change rank for cow-calf net income. As length of breeding season increased, more calves were born. As culling policies became more lax, culling rates decreased and more weanling calves were sold since fewer replacement heifers were needed.
7.3.1.7 Year Effects

Although year influenced many variables in VBCSM to some extent, the effects were significant only for culled cow revenue, cow-calf net income, stocker net income, culling rate and their correlated variables. Such effects are anticipated to some extent since the sales prices per hundredweight that were received for the various classes of simulated cattle in testing VBCSM varied between simulation years as reported earlier in this chapter. Year also, however, dramatically influenced the rate at which cows were culled, which was not anticipated. Culling rate, like culled cow revenue and cow-calf net income, was importantly influenced by year (Table 7-4). These large changes probably resulted from the way in which the age distribution of the original breeding herd was determined (normally distributed; Mean = 1730d, SD = 275d), the fact that aged cows were culled as they reached a maximum age of 15 years, or a combination of both.

Schons et al. (1985) characterized cattle populations in Virginia by constructing a life table from age at disposal data that were pooled over birth years, farms, and breeds. These data were Virginia Beef Cattle Improvement Association records on Angus, Hereford and Shorthorn females between 1939 and 1961 (Krehbiel et al., 1962). A more realistic age distribution in the foundation herd for VBCSM
simulations might have been achieved if each female created had been assigned an age by using this life table instead of the current method that assigned each female an age from a normal distribution (Mean = 1730d, SD = 275d).

Currently each cow is marked for culling from the breeding herd on the day that she reached a particular user-defined age that is referred to by the simulation as the culling age. The culling age may range from seven to 15 years but was set at 15 years in these simulations. If the animal was not lactating, it was culled on the same day that it was marked for culling. Otherwise, it was culled as soon as its calf was weaned or died. Again, the simulation might be more realistic if cows were culled for age only once a year on the day after their calves were weaned.

7.3.2. Descriptive Statistics of Biological Traits

Neither the calving season nor any of its interactions influenced the variables presented in Table 7-5. In fact, there appeared to be little difference between spring and fall calving systems for these biological traits, except for the size of the breeding herd at year-end. This suggests that the program code was acting in a consistent manner across seasons. While the size of the breeding herd (Number of Cows at Year-end, All Ages) differed between the two seasons, the difference between the two groups probably represented those lactating cows that had been marked for
culling but remained within the herd until their current calf was weaned. The average spring calf is born in April and weaned in November at an average age of 199 days, while the average fall calf is born in October and weaned in June at an average age of 239 days. Fall calves were still suckling their dams at year-end while spring calves were not.

The cattle production statistics report allows the dystocia rate for virgin heifers to be reported in the spring but not in the fall. The spring results were, however, lower than the average of the dystocia rates reported by Laster et al. (1973) and those reported by Meadows et al. (1994) for natural service calving ease bulls selected by visual inspection or by low accuracy expected progeny differences (EPD). Nonetheless, they are probably consistent with current expected dystocia rates since an adjustment was also made to compensate for the weight of contemporary cows compared to that of the average cow in 1973 when Laster and his associates conducted their studies. Some survey data suggest that dystocia rates in virgin heifers may be even lower than the dystocia rates reported for two year olds in Table 7-5 (USDA, 1995). Meadows et al. (1994) found dystocia rates to vary between three and 18% when virgin heifers were mated to bulls that were selected on the basis of high accuracy, low birth weight EPD’s. Perhaps the VBCSM should be modified to allow the use of
bulls selected on the basis of calving ease. If bulls were selected for calving ease, then dystocia rates reflecting the average results reported by Meadows et al. (1994) for cows mated to bulls with high accuracy calving ease EPD’s and those mated to bulls selected by either visual inspection or low accuracy EPD’s for calving ease could be used to calculate dystocia for 2 year olds in VBCSM. After adjusting for the size of contemporary cows, the results would probably not differ from dystocia rates that were reported by Meadows et al. (1994) when high accuracy calving ease EPD bulls were used.

7.4. Conclusions

Figures 7-4, 7-5, and 7-6 depict response surfaces of average cow-calf, stocker, and total net incomes, respectively, observed for each possible production strategy for the assumptions used to test VBCSM. For producers selling weanling calves (Figure 7-4), fall calving systems were consistently more profitable than spring calving systems, the 45 day breeding season was undesirable, and lax culling policies appear to be more profitable than stringent ones. For producers selling stocker calves (Figure 7-5), spring calved stockers were always more profitable than fall calved stockers. In fact, fall calved stockers were break-even at best. Again, the more lax culling policies appear to be more profitable than stringent ones but the 90 day
breeding season was more desirable than 45 or 60 day breeding seasons. Overall, when the stocker calf program was integrated into the cow-calf operation (Figure 7-6), much of the difference in financial results between spring and fall calving systems was canceled out. It is interesting to point out that there was very little increase in financial rewards by using a 90 day breeding season instead of a 60 day breeding season. The results depicted may not be valid for assumptions other than those discussed.

7.4.1. Potential Source Code Weaknesses

After examining VBCSM test simulations by using the three main effects and all their interactions of a 2 x 3 x 4 factorial model, the VBCSM appears to be working in a consistent and accurate fashion. However, several potential weaknesses in this model were identified. Although the VBCSM is a complex model, the addition of more detail to some of its biological functions would probably increase the model’s accuracy. The problem is that any increase in complexity must be balanced against the necessity of keeping the model as simple as possible in order to understand why it responds in a certain way as changes are made in the assumptions under which it works. The result would be that as the model becomes more complex, it would require additional assumptions whose accuracy may also be suspect.
7.5. Implications

After mathematically examining the VBCSM, the experimental results had several ramifications for Virginia cattle production. Among these were:

1. Fall calving appears to generate higher cow-calf net income to an enterprise than spring calving in Virginia.

2. Retaining spring calves through a stocker growing phase may be more profitable than retaining fall calves.

3. Spring calving in combination with retaining the spring calves as grass-fed stocker calves may be a profitable alternative to fall calving.

4. Virginia cattlemen may enhance their profits by more efficient utilization of production resources and advanced market planning.
Literature Cited


Figure 7-1a. Calving Season x Length Of Breeding Season Effects On Weanling Steer and Heifer Revenue.
a. Cow-Calf Net Income

Figure 7-1b. Calving Season x Length Of Breeding Season Effects On Cow-Calf Net Income, Stocker Calf Net Income, and Total Net Income.
b. Stocker Calf Net Income

Figure 7-1b (cont). Calving Season x Length Of Breeding Season Effects On Cow-Calf Net Income, Stocker Calf Net Income, and Total Net Income.
Figure 7-1b (cont). Calving Season x Length Of Breeding Season Effects On Cow-Calf Net Income, Stocker Calf Net Income, and Total Net Income.

c. Total Net Income
a. Heifer Revenue

Figure 7-2. Calving Season x Culling Policy Effects On Heifer Revenue, Stocker Heifer Revenue, Cow-Calf Net Income and Stocker Net Income.
b. Stocker Heifer Revenue

Figure 7-2 (cont).  Calving Season x Culling Policy Effects On Heifer Revenue, Stocker Heifer Revenue, Cow-Calf Net Income and Stocker Net Income.
c. Cow-Calf Net Income

Figure 7-2 (cont). Calving Season x Culling Policy Effects On Heifer Revenue, Stocker Heifer Revenue, Cow-Calf Net Income and Stocker Net Income.
Figure 7-2 (cont). Calving Season x Culling Policy Effects On Heifer Revenue, Stocker Heifer Revenue, Cow-Calf Net Income and Stocker Net Income.
Figure 7-3a. The Influence Upon Cow-Calf Net Income From The Interaction Between Length Of Breeding Season And Culling Policy.
Figure 7-3b. The Influence Upon Culling Rate From The Interaction Between Length Of Breeding Season x Culling Policy.
Figure 7-4. A Response Surface Displaying Predicted Cow-Calf Net Income From All Possible Management Combinations Of Calving Season, Length Of Breeding Season, And Culling Policy.
Figure 7-5. A Response Surface Displaying Predicted Stocker Calf Net Income From All Possible Management Combinations Of Calving Season, Length Of Breeding Season, And Culling Policy.
Figure 7-6. A Response Surface Displaying Predicted Total Net Income From All Possible Management Combinations Of Calving Season, Length Of Breeding Season, And Culling Policy.
Table 7-1. Important Calving Season Effects.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>FALL</th>
<th>SPRING</th>
<th>SE</th>
<th>P &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steer Revenue ($)</td>
<td>14825</td>
<td>11203</td>
<td>119</td>
<td>.0001</td>
</tr>
<tr>
<td>Heifer Revenue ($)</td>
<td>4291</td>
<td>2514</td>
<td>148</td>
<td>.0001</td>
</tr>
<tr>
<td>Cow-Calf Net Income ($)</td>
<td>5837</td>
<td>1613</td>
<td>144</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Steer Revenue ($)</td>
<td>18127</td>
<td>17530</td>
<td>143</td>
<td>.0032</td>
</tr>
<tr>
<td>Stocker Heifer Revenue ($)</td>
<td>5486</td>
<td>6045</td>
<td>200</td>
<td>.0486</td>
</tr>
<tr>
<td>Stocker Net Income ($)</td>
<td>(97)</td>
<td>3000</td>
<td>28</td>
<td>.0001</td>
</tr>
<tr>
<td>Total Net Income ($)</td>
<td>5648</td>
<td>4579</td>
<td>150</td>
<td>.0001</td>
</tr>
<tr>
<td>Weaning Weight (Lbs.)</td>
<td>563</td>
<td>479</td>
<td>.7</td>
<td>.0001</td>
</tr>
<tr>
<td>Average Sale Weight, Stockers (Lbs.)</td>
<td>783</td>
<td>675</td>
<td>.8</td>
<td>.0001</td>
</tr>
<tr>
<td>Cow-Calf Hay Consumption (Tons)</td>
<td>66.7</td>
<td>37.9</td>
<td>.6</td>
<td>.0001</td>
</tr>
<tr>
<td>Cow-Calf Concentrate Consumption (Cwt.)</td>
<td>96.7</td>
<td>.5</td>
<td>.7</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Hay Consumption (Tons)</td>
<td>.02</td>
<td>11</td>
<td>.2</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Concentrate Consumption (Cwt.)</td>
<td>.06</td>
<td>72</td>
<td>1.1</td>
<td>.0001</td>
</tr>
<tr>
<td>Cows Eligible For Breeding</td>
<td>92</td>
<td>87</td>
<td>.3</td>
<td>.0001</td>
</tr>
</tbody>
</table>
Table 7-2. Important Length Of Breeding Season Effects.

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>45 DAYS</th>
<th>60 DAYS</th>
<th>90 DAYS</th>
<th>SE</th>
<th>P &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steer Revenue ($)</td>
<td>12237</td>
<td>13502</td>
<td>13308</td>
<td>146</td>
<td>.0001</td>
</tr>
<tr>
<td>Heifer Revenue ($)</td>
<td>2287</td>
<td>3490</td>
<td>4431</td>
<td>181</td>
<td>.0001</td>
</tr>
<tr>
<td>Cow-Calf Net Income ($)</td>
<td>2679</td>
<td>3919</td>
<td>4576</td>
<td>177</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Steer Revenue ($)</td>
<td>16623</td>
<td>18341</td>
<td>18523</td>
<td>175</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Heifer Revenue ($)</td>
<td>3702</td>
<td>5867</td>
<td>7727</td>
<td>245</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Net Income ($)</td>
<td>1114</td>
<td>1395</td>
<td>1847</td>
<td>35</td>
<td>.0001</td>
</tr>
<tr>
<td>Total Net Income ($)</td>
<td>3796</td>
<td>5224</td>
<td>6321</td>
<td>184</td>
<td>.0001</td>
</tr>
<tr>
<td>Culling Rate (%)</td>
<td>26</td>
<td>23</td>
<td>22</td>
<td>.5</td>
<td>.0001</td>
</tr>
<tr>
<td>Pregnancy Rate (%)</td>
<td>78</td>
<td>82</td>
<td>86</td>
<td>.3</td>
<td>.0001</td>
</tr>
<tr>
<td>Weaning Rate (%)</td>
<td>64</td>
<td>68</td>
<td>73</td>
<td>.4</td>
<td>.0001</td>
</tr>
<tr>
<td>Weaning Weight, Lbs</td>
<td>539</td>
<td>524</td>
<td>499</td>
<td>.9</td>
<td>.0001</td>
</tr>
<tr>
<td>Average Age At Weaning, Days</td>
<td>226</td>
<td>221</td>
<td>210</td>
<td>.6</td>
<td>.0001</td>
</tr>
<tr>
<td>Number of Stockers Sold</td>
<td>38</td>
<td>47</td>
<td>52</td>
<td>1</td>
<td>.0001</td>
</tr>
</tbody>
</table>
Table 7-3. Important Culling Policy Effects.

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>CULLING POLICY 0</th>
<th>CULLING POLICY 1</th>
<th>CULLING POLICY 2</th>
<th>CULLING POLICY 3</th>
<th>SE</th>
<th>P &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steer Revenue ($)</td>
<td>13770</td>
<td>12435</td>
<td>13113</td>
<td>12740</td>
<td>168</td>
<td>.0001</td>
</tr>
<tr>
<td>Heifer Revenue ($)</td>
<td>3317</td>
<td>1928</td>
<td>4098</td>
<td>4268</td>
<td>209</td>
<td>.0001</td>
</tr>
<tr>
<td>Culled Cow Revenue ($)</td>
<td>9511</td>
<td>10333</td>
<td>10210</td>
<td>12214</td>
<td>318</td>
<td>.0001</td>
</tr>
<tr>
<td>Cow-Calf Net Income ($)</td>
<td>3158</td>
<td>2525</td>
<td>3808</td>
<td>5409</td>
<td>204</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Steer Revenue ($)</td>
<td>18434</td>
<td>16938</td>
<td>17928</td>
<td>18016</td>
<td>202</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Heifer Revenue ($)</td>
<td>5311</td>
<td>2827</td>
<td>6859</td>
<td>8064</td>
<td>283</td>
<td>.0001</td>
</tr>
<tr>
<td>Stocker Net Income ($)</td>
<td>1457</td>
<td>1165</td>
<td>1487</td>
<td>1698</td>
<td>40</td>
<td>.0001</td>
</tr>
<tr>
<td>Total Net Income ($)</td>
<td>4582</td>
<td>3634</td>
<td>5220</td>
<td>7018</td>
<td>212</td>
<td>.0001</td>
</tr>
<tr>
<td>Culling Rate</td>
<td>23%</td>
<td>28%</td>
<td>21%</td>
<td>23%</td>
<td>.5</td>
<td>.0001</td>
</tr>
<tr>
<td>% Pregnant</td>
<td>83%</td>
<td>84%</td>
<td>81%</td>
<td>79%</td>
<td>.3</td>
<td>.0001</td>
</tr>
<tr>
<td>Weaning Rate</td>
<td>70%</td>
<td>71%</td>
<td>67%</td>
<td>65%</td>
<td>.5</td>
<td>.0001</td>
</tr>
<tr>
<td>Replacement Heifers</td>
<td>25</td>
<td>25</td>
<td>21</td>
<td>21</td>
<td>1</td>
<td>.0002</td>
</tr>
<tr>
<td>Average Cow Age (Years)</td>
<td>4.9</td>
<td>4.1</td>
<td>6.1</td>
<td>7.8</td>
<td>.1</td>
<td>.0001</td>
</tr>
<tr>
<td>Number of Stockers Sold</td>
<td>46</td>
<td>37</td>
<td>48</td>
<td>51</td>
<td>1</td>
<td>.0001</td>
</tr>
</tbody>
</table>
Table 7-4. Important Year Effects On Selected Incomes and Traits By Years 1 - 3.

<table>
<thead>
<tr>
<th>Object</th>
<th>YEAR 1</th>
<th>YEAR 2</th>
<th>YEAR 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean Value</td>
<td>SE</td>
<td>Mean Value</td>
</tr>
<tr>
<td>Culled Cow Revenue</td>
<td>$ 6611</td>
<td>$ 147</td>
<td>$ 11487</td>
</tr>
<tr>
<td>Cow-Calf Net Income</td>
<td>$ 2287</td>
<td>$ 187</td>
<td>$ 4671</td>
</tr>
<tr>
<td>Stocker Net Income</td>
<td>$ 1458</td>
<td>$ 49</td>
<td>$ 1378</td>
</tr>
<tr>
<td>Culling Rate %</td>
<td>17.9</td>
<td>.4</td>
<td>24.6</td>
</tr>
</tbody>
</table>

P > F: .0001
### Table 7-5. Descriptive Statistics Of Selected Biological Traits By Calving Season.

<table>
<thead>
<tr>
<th>Object</th>
<th>FALL Mean Value</th>
<th>σ</th>
<th>SPRING Mean Value</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birth Weight (Lbs.)</td>
<td>89.6</td>
<td>1.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADG (Lbs.), Weanling Calves</td>
<td>1.98</td>
<td>.06</td>
<td>1.95</td>
<td>.07</td>
</tr>
<tr>
<td>ADG (Lbs.), Stocker Calves</td>
<td>1.8</td>
<td>.04</td>
<td>1.7</td>
<td>.04</td>
</tr>
<tr>
<td>Death Rate, Cows (%)</td>
<td>2.2</td>
<td>1.4</td>
<td>2.1</td>
<td>1.4</td>
</tr>
<tr>
<td>Death Rate, Calves (%)</td>
<td>7.3</td>
<td>3.2</td>
<td>6.9</td>
<td>3.1</td>
</tr>
<tr>
<td>*Dystocia Rates, 2 Year Olds (%)</td>
<td>---</td>
<td>---</td>
<td>32.3</td>
<td>19.8</td>
</tr>
<tr>
<td>Age Of Average Herd Cow (Years)</td>
<td>5.6</td>
<td>2.0</td>
<td>5.9</td>
<td>2.3</td>
</tr>
<tr>
<td>Number Of Cows at Year-End, All Ages</td>
<td>100</td>
<td>4</td>
<td>92</td>
<td>8</td>
</tr>
<tr>
<td>2 Year Olds w/Dystocia</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3 Year Olds w/Dystocia</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4 Year Olds &amp; Older w/Dystocia</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Eligible, Nonpregnant Cows</td>
<td>18</td>
<td>6</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Replacement Heifers Selected</td>
<td>24</td>
<td>15</td>
<td>22</td>
<td>6</td>
</tr>
<tr>
<td>Number Calves Born</td>
<td>68</td>
<td>10</td>
<td>67</td>
<td>9</td>
</tr>
<tr>
<td>Stockers Sold</td>
<td>45</td>
<td>12</td>
<td>46</td>
<td>12</td>
</tr>
</tbody>
</table>
CHAPTER 8. Summary and Implications

8.1. Introduction

8.2. Summary

8.3. Implications

Literature Cited
Chapter 8

Summary and Implications

Abstract: The Virginia Beef Cattle Simulation Model is written in C++ and is composed of five modules. It is a unique bio-economic cattle production simulation program that contains a forage module. It was tested by replicated simulations of two calving seasons, three breeding season lengths, and four culling policies. In addition to a management decision-aid capability, it has many other potential uses such as, for example, a classroom teaching tool.

8.1. Introduction

The purpose of this chapter is to summarize the Virginia Beef Cattle Simulation Model. This will be accomplished by providing a short review of this model and discussing its implications for future use and research.

8.2. Summary

The Virginia Beef Cattle Simulation Model (VBCSM) is a stochastic beef cattle production simulation program written in the C++ high level programming language. It consists of five interactive modules: reproduction and life events, forage, nutrition, marketing, and a tools module that contains the source code for the random seed, normal distributions, and procedures to convert days within calendar month to the appropriate day of the year. The purpose of VBCSM is to simulate biological and financial
inputs and outputs of real-world beef production for both cow-calf and stocker calf operations. It is based upon a set of assumptions in the form of a mathematical model that attempts to predict outcomes from changes in the management of various beef cattle production systems and their biological constraints with some degree of confidence. The VBCSM does this by allowing changes to be made in its input values. Then, by providing quantitative information, it can be determined if the outcomes vary. It is a flexible tool that can allow its user the opportunity to identify, evaluate, and compare the strengths, weaknesses, and consequences of different production and marketing alternatives before making a real-world commitment.

In the reproduction and life events module, many events are modeled as stochastic processes in order to introduce a statistical uncertainty about their success. Because reproduction as well as production outcomes contribute to financial success or failure, VBCSM uses many of these variables in a stochastic manner in order to incorporate variance into all classes of cattle. This allows management decisions to be based upon both biological responses and economics. Thus, VBCSM utilizes 23 stochastic variables including conception, time of parturition, postpartum interval, birth weight, potential mature weight and average daily gain. Since differences in weight are also a reflection of the amount and type of heterosis, VBCSM allows
average daily gain and birth weight be influenced by both direct and maternal heterosis. VBCSM tracks heterosis in crossbred animals for up to five generations. It allows the user to choose a breed or breed combination for the typical cow in its breeding herd from 12 common Virginia beef cattle breeds. The breed of the sire to be mated is also chosen from 12 beef breeds common to Virginia and may be different for virgin heifers than for mature cows.

Forage availability must be modeled before any beef production simulation system can accurately mirror the production decisions that confront beef cattle producers (Angirasa et al., 1981). Any failure to accurately model the intact beef cattle operating system results in simulation models that give inappropriate responses to input changes (Denham & Spreen, 1986). The VBCSM is unique among other beef production simulation models, including those specifically designed for research, because it reflects forage availability. This allows VBCSM to incorporate the dynamic interactions between forage growth and animal intake, including both the quantity and quality of the forages. Because forage availability reflects competition between cattle of the same or different classes, decisions concerning the stocking rate for the various classes of cattle as well as the length of time to retain stocker calves can be accurately portrayed. Grazing strategies may be devised by designing pastures from 21 forage compositions.
found in Virginia.

Its marketing module allows the economic consequences of changes in its biological systems to be measured to give the VBCSM a bio-economic quality. This gives the VBCSM the ability to study the optimal time and marketing strategy to sell calves and cull cows, whether or not replacement heifers should be purchased, if changes should be made in the breeding season, and many other management alternatives in terms of a cash flow analysis.

There are no hard and fast rules dictating the procedures to be used to examine a simulation model (Baldwin, 1992). VBCSM was examined by conducting test simulations and evaluating them by using a mathematical model with two classes for calving season, three classes for length of breeding season, four classes for culling policy, a year effect and all interactions. The year effect was analyzed using a time series analysis. The analysis gave VBCSM a rigorous examination and identified its general operating characteristics. Consistent results were obtained and potential weaknesses identified.

8.3. Implications

The VBCSM is a flexible, unique beef cattle bio-economic simulation model that is specifically designed as a management tool for Virginia cattle producers to compare economic ramifications of alternative production and
marketing strategies. Otherwise, because of the elements of time and risk, such testing could not be done. VBCSM is currently the only beef cattle simulation model that not only incorporates a forage module but allows all calculations to be made at the individual animal level. Because of its uniqueness and flexibility, it may also be used as classroom tool by educators to guide their students through the complicated, intricate high risk interactions which beef cattle producers face every day during the beef production process and predict the effects of their decision-making. Since the VBCSM is written in C++, a high level, objected-oriented programming language, its structure is encoded in a modular manner rather than in a procedural manner. This imparts an enhanced capacity for VBCSM to tolerate changes and modifications. This, in turn, allows VBCSM to be converted to a research model or adapted for use in other beef producing states. VBCSM’s usefulness may be further expanded by including a backgrounding module or a feedlot module to it or even converting it for use with another species. Finally, its user-friendly, interactive design allows the model to examine and evaluate changes in different management variables in order to observe how they fit in with other areas of animal science.
LITERATURE CITED


APPENDIX A
Figure A-2. The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-3. The First Dialog Page Of "Input Herd Values" Prompt Of The Main Menu Page In The Virginia Beef Cattle Simulation Model.
Figure A-4. The Second Dialog Page Of The “Input Herd Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
NOTE: THIS PROGRAM DEFINES STOCKER CATTLE AS POST-WEANLING GRASS FED CATTLE SUPPLEMENTED TO MEET NUTRITIONAL SHORT-FALLS.

Number of days to retain stocker calves: 120

Shall the simulation program calculate the post-weaning ADG for the Stocker Calves? Yes

Would you rather use an annual anticipated ADG instead of a seasonal anticipated ADG? Yes

Input the anticipated annual average daily gain, lbs/day? 1.2

Input the anticipated seasonal average daily gain for January 1 through March 31, lbs/day: 1

Input the anticipated seasonal average daily gain for April 1 through June 30, lbs/day: 1.5

Input the anticipated seasonal average daily gain for July 1 through September 30, lbs/day: 1.2

Input the anticipated seasonal average daily gain for October 1 through December 31, lbs/cow: 1

---

Figure A-5. The Third Dialog Page Of The “Input Herd Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-6. The Fourth Dialog Page Of The “Input Herd Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-7. The Fifth Dialog Page of the "Input Herd Values" Prompt of the Main Menu of the Virginia Beef Cattle Simulation Model.
Figure A-8. The First Dialog Page Of The “Input Forage Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-9. The Second Dialog Page Of The “Input Forage Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-10. The Third Dialog Page Of The “Input Forage Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-11. The Fourth Dialog Page Of The “Input Forage Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-12. The First Dialog Page Of The “Input Nutrition Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-13. The Second Dialog Page Of The “Input Nutrition Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-14. The First Dialog Page Of The “Input Market Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
### Input Simulation Values

<table>
<thead>
<tr>
<th>Annual Charges: Cow-Call Operation</th>
<th>Stocker/Grower Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artificial Insemination Charges ($/service): $</td>
<td>55</td>
</tr>
<tr>
<td>Building and Fence Repairs ($/cow): $</td>
<td>7</td>
</tr>
<tr>
<td>Current Payroll Tax Rates (decimal proportion):</td>
<td>0.0765</td>
</tr>
<tr>
<td>Implants ($/each):</td>
<td>1.0</td>
</tr>
<tr>
<td>Livestock Supplies ($/cow): $</td>
<td>2</td>
</tr>
<tr>
<td>Non-Crop Machinery Charges ($/cow): $</td>
<td>10</td>
</tr>
<tr>
<td>Pasture Maintenance Cost ($/acre): $</td>
<td>12.97</td>
</tr>
<tr>
<td>Salt and Mineral Cost ($/cwt of salt minerals): $</td>
<td>8</td>
</tr>
<tr>
<td>Cattle Cow Market Transportation Cost ($/cwt): $</td>
<td>5.2</td>
</tr>
<tr>
<td>Market Calf Transportation Cost ($/calf): $</td>
<td>4.5</td>
</tr>
<tr>
<td>Utilities ($/calf): $</td>
<td>1.25</td>
</tr>
<tr>
<td>Labor Charges ($/hour): $</td>
<td>8</td>
</tr>
<tr>
<td>Labor hours/cow)</td>
<td>8</td>
</tr>
<tr>
<td>Veterinary Services and Medicine ($/cow): $</td>
<td>12.25</td>
</tr>
<tr>
<td>Local Bank Interest Rates, Cow-Calf Operation: %</td>
<td>9.5</td>
</tr>
</tbody>
</table>

Figure A-15. The Second Dialog Page Of The “Input Market Values” Prompt Of The Main Menu Of The Virginia Beef Cattle Simulation Model.
Figure A-16. The "Continue" Prompt That Is Displayed After The Presentation Of The YEAR No. 1 And No. 2 Financial And Herd Summaries Of The Virginia Beef Cattle Simulation Model.
APPENDIX B
FUNCTIONS

Reproduction and Life Events

Cow
   A module that contains the functions and the
   beginning values of variables associated
   with the reproductive and life events portion
   of the simulation program.

Cow::Abort
   A function that causes a pregnant female
   to abort if certain physiological conditions
   are met.

Cow::AddAnimal
   A function that adds a new animal record to
   the linked list.

Cow::GetAnimal
   A function that retrieves the record of a given
   animal when needed for mother-calf functions.

Cow::GetHeadAnimal
   A function that calls the head animal on the
   linked list when it is necessary to count
   through the herd.

Cow::herdhead
   A function that retrieves the animal record at
   The head of the linked list. It is called by
   other functions that count through the herd.

Cow::BirthAnimal
A function that assigns the appropriate phenotypic and genetic characteristics to a fetus as soon as it is conceived by its dam.

Cow::BreedFemales
A function that gives breeding age females, regardless of parity, the opportunity to conceive.

Cow::CalfMilk
A function that assigns a calf the milk produced daily by its dam.

Cow::ChangeSex
A function that changes (1) the sex designation of weanling calves to their stocker calf designations and (2) the sex designations of replacement heifers to that of breeding cows when they reach puberty.

Cow::CreateHerd
A function that creates the original breeding herd.

Cow::CullCows
A function that culls breeding herd females in accordance with the designated culling policy.

Cow::DeadCattle(AnimalRecord *)
A function that calls the DeleteAnimal function when it is necessary to delete animals marked to die, or to delete the fetus of any pregnant
cow selected to be culled.

Cow::DeleteAnimal

A function that deletes the animal from the linked list after its death, abortion, culling, or sale.

Cow::DeleteHerd

A function deletes the herd when the program is finished.

Cow::DumpList

A function used in debugging.

Cow::Dystocia

A function that determines if a pregnant female experiences dystocia.

Cow::Init

A function whose purpose is to initialize the appropriate variables at the beginning of the simulation.

Cow::Lactation

A function that ensures that all lactating cows cease lactation after their calf is weaned.

Cow::LastID

A function that calls the last ID in use.

Cow::Mortality

A function that determines if an animal dies during the current day.

Cow::Parturition
A function that allows pregnant females to give birth to a live calf on the appropriate day.

Cow::PregnancyTest

A function that examines the pregnancy status of each breeding age female 60 days after the end of each breeding season.

Cow::Puberty

A function that determines when a replacement heifer calf exhibits its pubertal estrus.

Cow::ReproStats

A function that calculates the model’s annual reproductive statistics.

Cow::ResetCowVariables

This function resets annual counters to zero at year end.

Cow::ResetCowConCntr

This function resets the continuous counters in the cow module whenever the simulation is rerun without first exiting it.

Cow::SBBreeders

This function selects in-herd replacement heifers, purchases open replacement heifers, and purchases pregnant replacement heifers as necessary.

SBBreeders::GetSex

A local function in SBBreeders that sorts the
linked list for in-herd replacement heifers.

**Cow::ShrinkAnimal**
A function that calculates the shrunk and empty body weights in kilograms each day for each animal.

**Cow::Sterility**
A function that determines which breeding herd females become chronically unable to conceive or maintain a pregnancy longer than 16 days.

**Cow::UseInput1**
A function that allows the user to identify the farm.

**Cow::UseInput2**
A function that allows the user to input values for appropriate animal variables.

**Cow::UseInput2A**
A function that allows the user to specify breed composition of the breeding females and to specify the breed of the bull used in the herd.

**Cow::UseYearOption**
A function that allows the user to change the breed of the herd bull at the end of Year one and Year two.

**Cow::Wean**
A function that collects weaning statistics and
also counts number of cows in each age category at year end and body condition score category at the end of the last breeding season.

**Forage**

Forage

A function that contains all of the functions and the beginning values of the variables associated with the forage portion of the simulation program.

Forage::Init

This function initializes the appropriate forage variables at the beginning of the simulation.

Forage::MainForage

This function calculates the daily forage growth in pounds per acre per day for 21 forages common to Virginia after deducting for senescence and the previous day’s consumption.

Forage::ResetForageVariables

The purpose of this function is to reset all annual forage counters to zero at year end.

Forage::ResetForageConCntr

This function resets the continuous counters in the forage module whenever the simulation is
rerun without exiting.

Forage::Temperature

This function assigns the monthly random temperature occurring within the region in which the farm is located, based upon historical averages and standard deviations.

Forage::UseInput5

Allows the user to define forages specific to his personal situation.

Forage::UseInput6

This function ensures that the total forage composition of the pasture does not exceed 100%.

Forage::UseInput9

This function lets the user to input the number of pastures on the farm, the beginning and ending dates that they will be grazed and the beginning and ending dates during which they will be used to grow hay.

Forage::UseInput10

This function lets the user input the number of rental pastures on the farm, the beginning and ending dates that they will be grazed and the beginning and ending dates during which they will be used to grow hay.
Market

Market

A function that contains the functions and beginning values of variables associated with the marketing portion of the simulation program.

Market::BudgetReport

A function that calculates Net Returns from the cow-calf operation and Partial Budgets for the stocker calf operation during simulation years 1 - 3.

Market::Init

A function whose purpose is to initialize the marketing module of the simulation.

Market::PurchaseLivestock

A function that calculates the cost of replacement heifers that are purchased.

Market::ResetMarketVariables

A function that resets annual marketing counters to zero at year end.

Market::SellCullCows

A function that calculates culled cow revenues for the Income Statement.

Market::SellOrphans

A function that calculates orphan calf revenues for the Income Statement.
Market::SellStockerCalves
    A function that calculates stocker calf revenues for the Partial Budget.

Market::SellWeanlings
    A function that calculates weanling calf revenues for the Income Statement.

Market::UseInput3
    A function that allows the user to input data relative to his own business.

Market::UseInput4
    A function that allows the user to input data relative to his own business.

**Nutrition**

Nutrition
    A function that contains the functions and initial values of variables associated with the nutrition module of the simulation program.

Nutrition::DWGAIN
    This function predicts the daily weight gain (loss) of each animal based upon equations of the Nation Research Council (1996).

Nutrition::FeedUsage
    This function calculates annual cost of the supplemental feeds fed to the simulated herd, and
the net revenues of hay sold.

Nutrition::Init

A function that initializes values in the nutrition module.

Nutrition::MainNutrition

This function predicts the forage dry matter and supplemental feeds consumed by each animal based upon equations of the National Research Council (1996).

Nutrition::ResetNutritionVariables

This function resets all annual counters to zero at year end.

Nutrition::UseInput7

This function calculates the cost per pound of the supplemental feeds.

Nutrition::UseInput8

This function calculates the dry matter content, the metabolizable energy, the net energy for maintenance, and the net energy for gain of the supplemental feeds.

TOOLS

cleardevice

This function clears the computer screen.

GetRandom

This function generates 10,000 random numbers
after the random seed is generated.

GetStochasticNumber

This function returns a value from a normal distribution.

Julian

The function calculates the correct Julian calendar day.

VARIABLES

A1  The thermo-neutral maintenance requirement, Mcal/day/SBWTK.

A2  The maintenance adjustment for the previous month's ambient temperature, Mcal/day/SBWTK.

ABORT Indicates embryonic mortality.

ACBWT Indicates the average birth weight for any crossbred dam of a particular genetic composition including heterotic effects, if any.

ACDAT The average annual calving date, Julian calendar.

ACWF2 Average female calf weight for 2 year old dams.

ACWF3 Average female calf weight for 3 year old dams.

ACWF4 Average female calf weight for 4 year old and older dams.
<table>
<thead>
<tr>
<th>ACWM2</th>
<th>Average male calf weight for 2 year old dams.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACWM3</td>
<td>Average male calf weight for 3 year old dams.</td>
</tr>
<tr>
<td>ACWM4</td>
<td>Average male calf weight for 4 year old and older dams.</td>
</tr>
<tr>
<td>ADDR</td>
<td>The farm's mailing address.</td>
</tr>
<tr>
<td>ADGSCS</td>
<td>The average daily gain of stocker calves sold during the year.</td>
</tr>
<tr>
<td>ADGSH</td>
<td>The average daily gain of stocker heifers sold during the year.</td>
</tr>
<tr>
<td>ADGSS</td>
<td>The average daily gain of stocker steers sold during the year.</td>
</tr>
<tr>
<td>AdjFeedCost</td>
<td>The annual cost of supplement consumed by the cow-calf herd less the value of any farm hay applied to the cow-calf operation, dollars.</td>
</tr>
<tr>
<td>AdjSTFeedCost</td>
<td>The annual cost of supplement consumed by the stocker calves less the value of any farm hay applied to the stocker calf operation, dollars.</td>
</tr>
<tr>
<td>AGE</td>
<td>Age of the animal in days.</td>
</tr>
<tr>
<td>AgeBreeders[15]</td>
<td>An array containing the total number of breeding herd cows at year end that are of any age i, 1 – 15.</td>
</tr>
</tbody>
</table>
AGED
Age of the animal at death.

AgedCulls
The annual number of cows culled and sold for age restrictions.

AGFEE
The grading fee charged for cattle sold at auction that is included in sales fees.

AGRO
A variable storing the adjusted daily growth of the forage in any month, Kg DM/Acre.

AICS
Indicates artificial Insemination cost, $/cow.

AL
An intermediate rate constant used in calculating lactation energy requirements.

animal
The pointer for the current animal in the linked list.

animalPOINTER
A pointer to the first animal to be sorted when selecting potential in-herd replacements.

ANNRAIN
The estimated amount of rainfall in the year previous to the beginning of the simulation.

antiADG
The user-provided mean average daily post-weaning gain for the herd.

antiADG1
The user-provided mean average daily post-weaning gain for the herd from
January 1 through March 31.

antiADG2  The user-provided mean average daily post-weaning gain for the herd from April 1 through June 30.

antiADG3  The user-provided mean average daily post-weaning gain for the herd from July 1 through September 30.

antiADG4  The user-provided mean average daily post-weaning gain for the herd from October 1 through December 31.

APDMIF  The annual forage dry matter forage of the cow-calf operation, Kg.

APREP  Indicates the reduction (or gain) in forage yield due to monthly precipitation.

APRILGROWTH  The amount of forage growth occurring in April, Lbs/acre.

ARTI  Indicates if a breeding age female is bred by artificial insemination.

ASHKC  The average shrinkage of all culled cows sold, decimal proportion.

ASHKH  The average shrinkage of all heifers sold, decimal proportion.

ASHKH1  The average shrinkage of all heifers sold in YEAR 0 of the warm-up period, decimal proportion.
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASHKO</td>
<td>The average shrinkage of all orphan calves sold, decimal proportion.</td>
</tr>
<tr>
<td>ASHKS</td>
<td>The average shrinkage of all steer calves sold, decimal proportion.</td>
</tr>
<tr>
<td>ASHKS1</td>
<td>The average shrinkage of all steer calves sold in YEAR 0 of the warm-up period, decimal proportion.</td>
</tr>
<tr>
<td>ASHKSH</td>
<td>The average shrinkage of the stocker heifers sold, decimal proportion.</td>
</tr>
<tr>
<td>ASHKSS</td>
<td>The average shrinkage of the stocker steers sold, decimal proportion.</td>
</tr>
<tr>
<td>ASPDMIF</td>
<td>The annual dry matter forage intake for the stocker calf operation, Kg.</td>
</tr>
<tr>
<td>AUGUSTGROWTH</td>
<td>The forage growth occurring in August, Lbs/acre.</td>
</tr>
<tr>
<td>AvGestLen</td>
<td>A variable that indicates the average annual length of gestation for the pregnant cows in the breeding herd.</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>The average age of the breeding herd at year end.</td>
</tr>
<tr>
<td>AVFWEANAGE</td>
<td>The average weaning age of fall calving season calves.</td>
</tr>
<tr>
<td>AVGBWT</td>
<td>The average birthweight of calves born during the simulation year.</td>
</tr>
<tr>
<td>AVGBWTH</td>
<td>The average birthweight of heifer calves born during the simulation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>AVGBWTS</td>
<td>The average birthweight of steer calves born during the simulation year.</td>
</tr>
<tr>
<td>AvServConcp</td>
<td>The average number of services per conception across the breeding herd.</td>
</tr>
<tr>
<td>AVSWEANAGE</td>
<td>The average weaning age of spring calving season calves.</td>
</tr>
<tr>
<td>AWCC</td>
<td>The average weight of culled cows sold during the year.</td>
</tr>
<tr>
<td>AWHS</td>
<td>The average weight of the heifers sold during the year.</td>
</tr>
<tr>
<td>AWNWT</td>
<td>The average weaning weight of the calves weaned during the year.</td>
</tr>
<tr>
<td>AWSHS</td>
<td>The average weight of the stocker heifers sold.</td>
</tr>
<tr>
<td>AWSS</td>
<td>The average weight of the steer calves sold.</td>
</tr>
<tr>
<td>AWSSS</td>
<td>The average weight of the stocker steers sold.</td>
</tr>
<tr>
<td>BackupCow</td>
<td>A fictitious animal created to retain access to other animals in the linked list in the event that the current animal in the linked list dies or is otherwise deleted.</td>
</tr>
</tbody>
</table>
| BASS         | The Virginia basis for futures market
operations, $/cwt.

**BBWT[13]**  An array containing the average birth weights for bulls of a particular breed, 1 - 12.

**BCS**  The animal's Body Condition Score, a score ranging from 1 to 9 (1 = extremely thin; 9 = extremely fat) for body fatness.

**BCScore[10]**  An array containing the total number of breeding herd cows at year end with a body condition score of i, 1 - 9.

**BCST**  The cost of each bull used in the breeding herd.

**BDCC**  The breed premium/discount for any particular culled cow sold.

**BDEP**  The total annual depreciation charge for the breeding bulls.

**BDH**  The breed premium/discount for any particular heifer sold.

**BDOC**  The breed premium/discount for any particular orphan calf sold.

**BDS**  The breed premium/discount for any particular steer calf sold.

**BDSH**  The breed premium/discount for any particular stocker heifer sold.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDSS</td>
<td>The breed premium/discount for any particular stocker steer sold.</td>
</tr>
<tr>
<td>BFRC</td>
<td>The cost per cow of building and fence repairs for the cow-calf herd.</td>
</tr>
<tr>
<td>BFRP</td>
<td>The total cost of building and fence repairs charged to the cow-calf operation.</td>
</tr>
<tr>
<td>BFRPS</td>
<td>The total cost of building and fence repairs charged to the stocker calf operation.</td>
</tr>
<tr>
<td>BFRS</td>
<td>The cost per cow of building and fence repairs, $/head.</td>
</tr>
<tr>
<td>BIRTHDATE</td>
<td>Indicates the birthday of the calf.</td>
</tr>
<tr>
<td>BPMWT[13]</td>
<td>An array containing the average potential mature weights for</td>
</tr>
</tbody>
</table>
female offspring of bulls of a particular breed, 1 - 12.

**BREED**

A numerical code identifying the breed of a purebred cow or calf or the sire breed of a crossbred cow or calf: 1 = Angus; 2 = Hereford; 3 = Polled Hereford; 4 = Shorthorn; 5 = Charolais; 6 = Chianina; 7 = Gelbvieh; 8 = Limousin; 9 = Maine Anjou; 10 = Pinzgauer; 11 = Simmental; 12 = Tarentaise.

**BreedingCows**
The total number of breeding females of all ages in the herd including those selected for culling and as replacement breeding females.

**BreedingReplacement**
Indicates the number of unweaned heifers currently selected as breeding herd replacements.

**BSAL**
The estimated salvage value of a breeding bull relative to its original cost, decimal proportion.

**BSD**
The standard deviation of the birthweight of an animal of a particular genetic composition.

**BULBREED[13]**
An array that contains a numerical
designation identifying each bull breed.

BUSE The number of years that a bull will be kept in the breeding herd prior to culling.

BuyBreed The breed of replacement females that will be purchased.

BUYP Indicates if pregnant females will be purchased as herd replacements: 
0 = NO; 1 = YES.

BuyPure Indicates whether purebred or crossbred replacements will be purchased: 0 = a purebred animal; 1 = a crossbred animal.

BUYRC Indicates if open replacement females are to be purchased: 0 = NO; 1 = YES.

BuySex Indicates which sex of calves will be purchased to replace cattle selected for the breeding herd: 0 = Females; 1 = Males. Initially, it is a constant set to 0.

BWT The birth weight of the calf.

BWTAVG The average birthweight for an animal of a particular genetic composition.

CADAT The calving date, Julian calendar day.
calcuADG  A boolean variable indicating whether or not the simulation program is to calculate the post-weaning ADG for the animals or whether a user-provided value will be used for the mean ADG: 0 = Use the provided ADG; 1 = Calculate the post-weaning ADG.

calcuseasADG  A boolean variable indicating whether or not an annual user-provided value will be used for the mean ADG or a user-provided mean ADG for each season will be used for the mean ADG: 0 = Use annual user-provided mean post weaning ADG; 1 = the user-provided seasonal ADG.

CalfCows  The number of eligible cows in the previous year.

CalfCows1  The number of eligible cows in the previous year.

CALFEASE  Indicates if a bull selected on the basis of calving ease will be mated to heifers.

CALFSEAS  Indicates whether the calf was born during spring calving season or fall calving season.

CarryOver  Indicates the total forage available
for animal consumption in dry matter on January 1 of each year, Kg.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBIRTHDATE</td>
<td>The day that the animal's last calf was born.</td>
</tr>
<tr>
<td>CBWT</td>
<td>The birthweight of the animal’s calf.</td>
</tr>
<tr>
<td>CCRV</td>
<td>The revenue obtained from the sale of an individual culled cow.</td>
</tr>
<tr>
<td>CDMIEQ</td>
<td>The total daily dry matter intake of an unweaned calf expressed in a forage equivalent. It includes all forage, supplemental feeds and milk consumed by the suckling calf daily.</td>
</tr>
<tr>
<td>CDWT</td>
<td>The animal's current live weight in pounds.</td>
</tr>
<tr>
<td>CDWTK</td>
<td>The animal's current live weight in kilograms.</td>
</tr>
<tr>
<td>CGREDCost</td>
<td>The total cost per pound of the supplement fed to breeding cows, dollars/lb.</td>
</tr>
<tr>
<td>CIDN</td>
<td>Indicates the ID of any particular breeding cow's unweaned calf.</td>
</tr>
<tr>
<td>CLNUM</td>
<td>The number of females culled during the simulation year.</td>
</tr>
<tr>
<td>CNDAT</td>
<td>The conception date, Julian calendar day.</td>
</tr>
<tr>
<td>CNEM</td>
<td>The net energy required for the</td>
</tr>
</tbody>
</table>
maintenance of the animal.

CNEMAP  The animal activity effect on the net energy for maintenance if the animal is on pasture.

CNEMCS  Net energy for maintenance required by the animal due to cold stress, Mcal/day.

CNEMP   The net energy required for the maintenance of pregnant females.

CNRAT   The annual conception rate for the simulation year.

CNTR    The current size of a Chicago Mercantile Exchange Feeder Cattle futures contract.

CNUMB   A counter that records the total number of calves born during the simulation year.

CNUMBH  A counter that records the total number of heifers born during the simulation year.

CNUMBS  A counter that records the total number of steers born during the simulation year.

COLR    Indicates the animal's color:
        0 = Black; 1 = Not Black.

CONCE   The number of pregnant females.
CONCE1 The number of pregnant cows during the last breeding season for fall calving.
Concentrate Indicates the amount of concentrates consumed annually by the cow-calf herd.
ConcentrateS Indicates the amount of concentrates consumed annually by the stocker calf herd.
CONCV The probability that a female becomes pregnant during an estrus when bred.
CONUM The total number of breeding females of all ages in the herd plus one reserved animal that always remains in the linked list.
Correction A variable used to convert the day in any calendar month into its day number in the calendar year.
CostofReplacements A variable that is used in reporting the total cost of herd replacements purchased during the year.
CostSKCC The discount value in dollars for the shrinkage in any culled cows sold.
CostSKHC The discount value in dollars for the shrinkage in any heifers sold.
CostSKHC0 The discount value in dollars for the shrinkage in all heifers sold in
YEAR 0 of the warm-up period

CostSKOC The discount value in dollars for the shrinkage in any orphan calves sold.

CostSKSC The discount value in dollars for the shrinkage in any steers sold.

CostSKSCO The discount value in dollars for the shrinkage in all steers sold in YEAR 0 of the warm-up period.

CostSKSHC The discount value in dollars for the shrinkage in any stocker heifers sold.

CostSKSSC The discount value in dollars for the shrinkage in any stocker steers sold.

count A local counter variable used in the CreateHerd function.

COWS The total number of breeding females in the herd.

CowsAbort The number of pregnant cows that aborted.

CowsElig The number of cows in the herd eligible to breed before the end of breeding season.

CowsEligl The number of cows in the herd eligible to breed before the end of the last breeding season for fall calving.

CowsFail A counter that indicates the number of breeding cows in the herd that
failed to give birth to a calf.

**CowsFailWean**
A counter that indicates the number of breeding cows in the herd that gave birth to a calf that died before it was weaned.

**CowsInElig**
A counter that indicates the number of heifers in the herd that were ineligible to breed during the heifer breeding season.

**CowsInElig1**
The number of heifers in the herd that were ineligible to breed during the heifer breeding season for fall calving.

**CPDMI**
The total predicted dry matter intake for the forage plus any supplemental feeds, Kg.

**CPDMIS**
The predicted daily dry matter intake from supplemental feeds for cows, Kg.

**CPRSA1**
The estimated average price of a Virginia commercial grade cow in Year # 1 as specified by the user.

**CPRSA2**
The estimated average price of a Virginia commercial grade cow in Year # 2 as specified by the user.

**CPRSA3**
The estimated average price of a Virginia commercial grade cow in Year #
3 as specified by the user.

**CRATE**
The annual calving rate for the simulation year, percentage.

**CSAME**
The metabolizable energy of the supplement for the breeding herd, Mcal/Kg.

**CSANEG**
The available net energy for gain from the supplement for the breeding herd, Mcal/Kg.

**CSANEM**
The available net energy for maintenance from the supplemental feeds for the breeding herd, Mcal/Kg.

**CSEX**
Indicates the sex of an animal’s calf:
0 = Female; 1 = Male.

**CUCYC**
The length of the cumulative annual estrous cycles in days.

**CULAG**
The age at which females are automatically culled from the breeding herd: 8 = 8 years of age; 9 = 9 years of age; 10 = 10 years of age; 11 = 11 years of age; 12 = 12 years of age; 13 = 13 years of age; 14 = 14 years of age; 15 = 15 years or more in age.

**CULL**
Indicates if a cow is to be culled:
0 = Not culled; 1 = culled.

**CulledCows**
The number of breeding cows
selected for culling but not yet removed from the herd.

**CULPO**  
The herd culling policy specified:  
0 = cull open females after pregnancy testing; 1 = cull females failing to calve or to wean a calf in any one year; 2 = cull females failing to calve or to wean a calf in any 2 years; 3 = cull females failing to calve or to wean a calf in any 3 years.

**CULRAT**  
The culling rate for the simulation year, percentage.

**current**  
A variable used to sort the linked list.

**currentDATE**  
The current day of the Julian year.

**currentID**  
The identification number of the current animal in the linked list.

**CurrentSIZE**  
The number of acres of pasture dedicated to the cow-calf operation.

**CurrentSIZES**  
The number of acres of pasture dedicated to the stocker calf operation.

**CyclCow1**  
The number of cycling cows during the first third of the regular breeding season for spring calving.

**CyclCow2**  
The number of cycling cows during the
second third of the regular breeding season for spring calving.

CyclCow3 The number of cycling cows during the last third of the regular breeding season for spring calving.

CyclCow21 The number of cycling cows during the first third of the regular breeding season for fall calving.

CyclCow22 The number of cycling cows during the second third of the regular breeding season for fall calving.

CyclCow23 The number of cycling cows during the last third of the regular breeding season for fall calving.

CyclHeifer1 The number of cycling heifers during the first third of the heifer breeding season for spring calving.

CyclHeifer2 The number of cycling heifers during the second third of the heifer breeding season for spring calving.

CyclHeifer3 The number of cycling heifers during the last third of the heifer breeding season for spring calving.

CyclHeifer21 The number of cycling heifers during the first third of the heifer breeding season for fall calving.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyclHeifer22</td>
<td>The number of cycling heifers during the second third of the heifer breeding season for fall calving.</td>
</tr>
<tr>
<td>CyclHeifer23</td>
<td>The number of cycling heifers during the last third of the heifer breeding season for fall calving.</td>
</tr>
<tr>
<td>DAMBREED</td>
<td>Indicates the breed of the animal’s dam, 1 through 12 as listed.</td>
</tr>
<tr>
<td>DATE</td>
<td>The current day of the Julian year.</td>
</tr>
<tr>
<td>dateIN[7]</td>
<td>An array that contains the annual dates that cattle will be allowed to graze hay pastures.</td>
</tr>
<tr>
<td>dateOUT[7]</td>
<td>An array that contains the annual dates that cattle will be removed from each hay pasture.</td>
</tr>
<tr>
<td>dateRIN[7]</td>
<td>An array that contains the annual dates that stocker cattle will be allowed to graze rental hay pasture.</td>
</tr>
<tr>
<td>dateROUT[7]</td>
<td>An array that contains the annual dates that stocker cattle will be removed from each rental hay pasture.</td>
</tr>
<tr>
<td>DAY</td>
<td>The current simulation day.</td>
</tr>
<tr>
<td>DaySen</td>
<td>The total daily pasture senescence in lbs.</td>
</tr>
<tr>
<td>DEADC</td>
<td>The calf death rate based upon the</td>
</tr>
</tbody>
</table>
number of calves born during the year, decimal proportion.

DECEMBERGROWTH  The total amount of forage growth occurring in December, Lbs/acre.

DETEC  Indicates if the female's estrus period is detected: 0 = Estrus is not detected; 1 = estrus is detected.

DGRO  A variable used to store the daily growth of pasture forage in pounds per acre.

DHETADG  Indicates the increase in the calf's potential average daily gain (PADG) resulting from individual (direct) heterosis.

DHETBWT  Indicates the increase in the calf's potential birthweight (BWT) resulting from individual (direct) heterosis.

DHETMWT  Indicates the increase in the calf's potential mature weight (PMWT) resulting from individual (direct) heterosis.

DIE  Indicates if the animal dies: 0 = the animal lives; 1 = the animal dies.

DIPM  Daily initial pasture mass, Kg DM/Ha.

DIST  The distance to the closest auction
market or gathering yard of the Tel-O-Market to which the cattle will be taken, miles.

DMandSC The daily mineral and salt consumption of stocker calves, cwt./head/day.

DNUM Indicates the number of calves that died during the simulation year.

DNUMF Indicates the number of cows that died during the simulation year.

DNUMH Indicates the number of heifers that died during the simulation year.

DNUMHS Indicates the number of stocker heifers that died during the simulation year.

DNUMR Indicates the number of replacement heifers that died prior to weaning.

DNUMS Indicates the number of steer calves that died during the simulation year.

DNUMSC Indicates the total number of stocker calves that died during the simulation year.

DNUMSS Indicates the number of stocker steers that died during the simulation year.

DPREM The premium received for the direct sale of cattle, decimal proportion.

DPREG The number of days that the cow has
been pregnant.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAT</td>
<td>The cow herd death rate.</td>
</tr>
<tr>
<td>DRATS</td>
<td>The stocker (retained) calf death rate for the simulation year.</td>
</tr>
<tr>
<td>DRET</td>
<td>The number of days that stocker calves will be retained.</td>
</tr>
<tr>
<td>DYPerF2</td>
<td>The percent of 2-year-old dams carrying female calves that will experience dystocia.</td>
</tr>
<tr>
<td>DYPerF3</td>
<td>The percent of 3-year-old dams carrying female calves that will experience dystocia.</td>
</tr>
<tr>
<td>DYPerF4</td>
<td>The percent of 4-year-old or older dams carrying female calves that will experience dystocia.</td>
</tr>
<tr>
<td>DYPerM2</td>
<td>The percent of 2-year-old dams carrying male calves that will experience dystocia.</td>
</tr>
<tr>
<td>DYPerM3</td>
<td>The percent of 3-year-old dams carrying female calves that will experience dystocia.</td>
</tr>
<tr>
<td>DYPerM4</td>
<td>The percent of 4-year-old or older dams carrying female calves that will experience dystocia.</td>
</tr>
<tr>
<td>DYSTF</td>
<td>The dystocia factor. The calculated probability that dystocia occurs.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DYSTO</td>
<td>Indicates if a female will experience dystocia: 0 = the female will not exhibit dystocia during the current parturition; 1 = The female will exhibit dystocia during the current parturition.</td>
</tr>
<tr>
<td>Dystocia2</td>
<td>The number of females less than two and one half years of age experiencing dystocia.</td>
</tr>
<tr>
<td>Dystocia3</td>
<td>The number of females between two and one half and three and one half years of age experiencing dystocia.</td>
</tr>
<tr>
<td>Dystocia4</td>
<td>The number of females over three and one half years of age experiencing dystocia.</td>
</tr>
<tr>
<td>E</td>
<td>The energy content of the milk, Mcal (NEM)/kg.</td>
</tr>
<tr>
<td>earlyday</td>
<td>The first day of the heifer breeding season for spring calving.</td>
</tr>
<tr>
<td>earlyday2</td>
<td>The first day of the heifer breeding season for fall calving.</td>
</tr>
<tr>
<td>earlymonth</td>
<td>The month in which the first day of the heifer breeding season for spring calving begins.</td>
</tr>
<tr>
<td>earlymonth2</td>
<td>The month in which the first day of the heifer breeding season for fall calving begins.</td>
</tr>
</tbody>
</table>
calving begins.

**EBWTK**  
The animal's current empty body weight in kilograms.

**EligCowsOpen**  
A variable that indicates the number of eligible breeding cows in the herd that are not pregnant at the time of pregnancy testing.

**EPREM**  
The premium received for the Tel-O-Market auction sales of cattle, decimal proportion.

**EQSBW**  
The animal's equivalent shrunk body weight (an NRC term indicating that the animal's body weight has been adjusted to a standard reference weight) in kilograms.

**EQEBW**  
The animal's equivalent empty body weight (an NRC term indicating that the animal's body weight has been adjusted to a standard reference weight) in kilograms.

**EROS**  
Indicates the potential for erosion on the grazing unit: 1 = severe erosion potential; 2 = moderate erosion potential; 3 = light erosion potential; 4 = no erosion potential.
excessSBWTG The difference between the body weight calculated by the nutrition module when a stocker calf is fed to a calculated daily weight gain target and that calculated by the simulation when it has no specific target weight.

excessSPDMI The extra feed required to feed a stocker calf beyond a calculated daily weight gain target.

ExternInsul The animal's external insulation value, degrees Celcius/Mcal/m²/day.

extraRE The difference between the retained energy calculated by the nutrition module when a stocker calf is fed to a calculated daily weight gain target and that calculated by the simulation when it has no specific target weight.

EZBREED Indicates the sire breed to mate with the herd's heifers and heiferettes.

EZCALF Indicates if a different sire breed will be mated to the heifers and heiferettes than to the mature cows: 0 = No; 1 = Yes.

FAIL The number of times that a female has failed to bear a live calf.

FAME The metabolizable energy of the
pasture forage, Mcal/Kg.

FANEG  The net energy for gain of the pasture forage, Mcal/Kg.

FANEM  The net energy for maintenance of the pasture forage, Mcal/Kg.

FARM   The farm name.

FARMSIZE  The total size of the farm including rental pastures in acres.

FatCompFac  The effect of the previous plane of nutrition on the NEM requirement.

FAVAIL[22]  An array containing the amount of the potential standing cover for each forage at the beginning of the simulation, Lbs/acre.

FBORT  This indicates potential fetal mortality after 45 days of pregnancy and prior to parturition.

FCALF  The total number of fall calves weaned.

FDBS  The first day of the current year's breeding season for spring calving.

FDBS2  The first day of the breeding season for fall calving.

FDBHS  The first day of the current year's heifer breeding season for spring calving.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDBHS2</td>
<td>The first day of the heifer breeding season for fall calving.</td>
</tr>
<tr>
<td>fdbsdays</td>
<td>The first day of the breeding season for spring calving.</td>
</tr>
<tr>
<td>fdbsdays2</td>
<td>The first day of the breeding season for fall calving.</td>
</tr>
<tr>
<td>fdbsmonth</td>
<td>The month in which the first day of the breeding season for spring calving begins.</td>
</tr>
<tr>
<td>fdbsmonth2</td>
<td>The month in which the first day of the breeding season for fall calving begins.</td>
</tr>
<tr>
<td>FEBRUARYGROWTH</td>
<td>The total amount of forage growth in February, Lbs/acre.</td>
</tr>
<tr>
<td>FECA</td>
<td>The total fees charged for selling each weanling calf.</td>
</tr>
<tr>
<td>FECAS</td>
<td>The total fees charged for selling each stocker calf.</td>
</tr>
<tr>
<td>FEE</td>
<td>The total sales commissions for each calf sold, i.e., $3.00 + $1.00 check-off per head + .50 grading fee per head.</td>
</tr>
<tr>
<td>FEEA</td>
<td>The fee adjustment for each calf over 600 pounds and sold in an Tel-O-Market auction.</td>
</tr>
<tr>
<td>FEEC</td>
<td>The total sales commissions for each</td>
</tr>
</tbody>
</table>
culled cow sold, i.e. $ 4.00 + $ 1.00 check-off per head.

FeedCost  The annual cost of the supplement consumed by the cow-calf herd, dollars.

FERT  Indicates the level of pasture maintenance: 0 = low levels of maintenance; 1 = average levels of maintenance; 2 = high levels of maintenance.

FESTR  Indicates a fertile estrus.

Fgeneration  Indicates the generation number of a crossbred animal.

FICA  The current federal payroll tax rate, decimal proportion.

FIDN  The identification number of the fetus.

FMAN  The name of the farm manager.

FME[22]  An array that contains the metabolizable energy of each forage species, Kg.

FNEG[22]  An array containing the net energy for gain for each forage species, Mcal/Kg.

FNEM[22]  An array containing the net energy for maintenance for each forage
species, Mcal/Kg.

**FOWN**
The name of the farm owner.

**FR[22]**
An array storing the percentage of each forage species in the grazing unit:
1 = Fescue; 2 = Fescue-Ladino Clover; 3 = Kentucky Bluegrass;
4 = Kentucky Bluegrass/White Clover;
5 = Orchardgrass; 6 = Orchardgrass-Ladino Clover; 7 = Orchardgrass-Red Clover; 8 = Alfalfa; 9 = Bermudagrass 100N; 10 = Bermudagrass 200N; 11 = Bermudagrass 250N; 12 = Bermudagrass-Rye grass; 13 = Bahiagrass 200N; 14 = Dallisgrass; 15 = Kudzu; 16 = Lespedeza; 17 = Rye grass 150N; 18 = Sericea; 19 = Switchgrass 140N; 20 = Pearl Millet; 21 = Sudan Sorghum 150N.

**FSELLDATE**
The day that weanling calves born during the fall calving season are sold.

**FWEAN**
Indicates the number of seasons that a cow fails to wean a calf.

**GDAMBREED**
Indicates the breed of the animal’s maternal granddam, 1 – 12.

**GFEE**
The grading fee charged for each calf sold through the Virginia Tel-O-
GGDAMBR\textsc{eed} Indicates the breed of the animal’s maternal great-granddam, 1 – 12.

GGGDAMBR\textsc{eed} Indicates the breed of the animal’s maternal great, great-granddam, 1 – 12.

GGGSIRE\textsc{breed} Indicates the breed of the animal’s maternal great, great-grandsire, 1 – 12.

GGSIRE\textsc{breed} Indicates the breed of the animal’s maternal great-grandsire, 1 – 12.

GRAZ Describes the cattle-forage management system: 1 = continuous grazing; 2 = simple rotation; 3 = a complex rotational grazing system.

GRAZE The decimal proportion of the diet supplied by forage.

GRCost[21] An array storing the cost per pound of the supplemental ingredients, dollars/lb.

GREDCost The total cost per pound of the supplement, dollars/lb.

GSIRE\textsc{breed} Indicates the breed of the animal’s maternal grandsire, 1 – 12.

GUNIT The grazing unit, i.e., the pasture (farm) size in hectares.

GUWT\textsc{k} Indicates the current weight of the
gravid uterus including fluids, membranes, and fetus in kilograms.

GUWTKD The calculated daily weight gain of the gravid uterus during each day of the pregnancy.

GUWTKT The cumulative weight gain in the gravid uterus over the course of the pregnancy.

HAIR The effective hair depth of the animal's coat for insulation: .2 = May - October; .5 = October - May.

hayAVAIL Indicates the total hay available for animal consumption in dry matter, Lbs.

hayConsume Indicates the amount of hay consumed annually by the cow-calf herd.

HayConsumeS Indicates the amount of hay consumed annually by the stocker calf herd.

HAYCOST The cost of producing farm hay, dollars/ton.

HayCost The cost of producing farm hay in dollars/Lb.

hayFDM[22] An array that contains the dry matter content of forages that have been cut and cured for hay.

hayLANDS This indicates the total number of
acres on which hay will be grown during each month of the simulation year.

hayRevenue The revenue from farm grown hay.

hayRLANDS This indicates the total number of rental acres on which hay will be grown during each month of the simulation year.

hayVALUE The gross value of the hay in dollars/Lb.

HBIC Income/loss from hedging operations for the cow-calf operation.

HBIS Income/Loss from hedging operations for the stocker calf operation.

HBREED The user's inputted cow breed(s).

HBBREED The user's inputted sire breed.

HDCE The total cost of hedging operations for the cow-calf operation.

HDCS The total cost of hedging operations for the stocker operation.

HDEX The brokerage cost for each futures contract, $/contract.

head A local variable in the function, GetAnimal (int ID), that identifies each animal on the linked list as it becomes the head animal on the
HeatProd  The animal's daily body heat production, Mcal/day.

HEDG1  A boolean variable that indicates if the calves will be hedged: 0 = NO; 1 = YES.

HEDG2  A boolean variable that indicates if the stocker calves will be hedged: 0 = NO; 1 = YES.

herdhead  A local variable that retrieves the head animal on the linked list whenever a search is conducted.

HFRV  The marketing revenue from each heifer calf that is sold.

HIDE  The thickness of the animal's hide: 1 = thin; 2 = average (most European Breeds); 3 = Hereford/Hereford crossbred.

HLAND[7]  An array containing the number of acres in each pasture utilized for both grazing lands and hay lands depending upon the month of the simulation year.

HOUR  The amount of labor used for the cow-calf herd, hours/cow.

HOURS  The amount of labor used for the
stocker herd, hours/head.

**HRLAND[7]** An array containing the number of acres in each rental pasture used for both grazing lands and hay lands depending upon the month of the simulation year.

**HSEAS** Indicates if the heifer breeding season will start before the regular breeding season for spring calving:

0 = NO; 1 = YES.

**HSEAS2** Indicates if the heifer breeding season will start before the regular breeding season for fall calving:

0 = NO; 1 = YES.

**I** A looping counter.

**ID** The identification number of the animal.

**IM** The daily intake required for the animal's maintenance.

**IMPC** The total cost of implants charged to the cow-calf operation.

**IMPH** Indicates if the heifer calves are implanted prior to weaning.

**IMPS** Indicates if steer calves are implanted prior to weaning.

**IMPSC** The total cost of the implants used
for the stocker cattle operation.

**Insulate**
The animal's total insulation value, degrees Celcius/Mcal/m²/day.

**INSURS**
The unit cost of insurance for the stocker herd, $/head.

**INSUS**
The total cost of insurance for the stocker calf operation.

**INTE**
The interest on borrowed capital for the cow-calf business.

**InternInsul**
The animal's tissue (internal) insulation value, degrees Celcius/Mcal/m²/day.

**INTES**
The interest on capital borrowed for the stocker calf operation.

**INTR**
Current interest rates on cow-calf operating loans, decimal proportion.

**INTRS**
Current interest rates on stocker calf operating loans, decimal proportion.

**IONO**
Indicates if the animal is being given an ionophore.

**IPCS**
The cost of a implant.

**IMPH**
Indicates if heifer calves are implanted prior to weaning: 0 = NO; 1 = YES.
IMPLANT Indicates if the animal will be implanted: 0 = NO; 1 = Yes.

IMPS Indicates if steer calves are implanted prior to weaning: 0 = NO; 1 = YES.

IMPSS Indicates if a stocker steer is given implants: 0 = NO; 1 = YES.

IMPHS Indicates if a stocker heifer is given implants: 0 = NO; 1 = YES.

IONO Indicates if the animal is being given an ionophore: 0 = NO; 1 = YES.

j A local counter.

JANUARYGROWTH The total amount of forage growth in January, Lbs/acre.

JulianValue The Julian calendar day of the simulation year.

JULYGROWTH The total amount of forage growth in July, Lbs/acre.

JUNEGROWTH The total amount of forage growth in June, Lbs/acre.

KL An rate constant used in calculating the energy requirements for lactating females.

KM The ratio of the diet NEM to diet ME.

LABO The total labor charges for the cow-calf operation.
LABOS  The total labor charges for the stocker calf operation.

LACT  This indicates if the female is lactating:  0 = NO; 1 = YES,

LASS  The local cattle association fee charged for each calf sold by Tel-O-Market auction.

lastDAY  The last day of the simulation month.

LDAY  A counter that records the number of consecutive days that a female has been lactating.

LDBS  The last day of the current year's regularly scheduled breeding season for spring calving.

LDBS2  The last day of the current year's scheduled breeding season for fall calving.

LMAN  The name of the livestock manager.

LORT  The size of operating loans borrowed for the cow-calf operation as a proportion of estimated operating expenses, decimal proportion.

LORTS  The size of operating loans borrowed for the stocker herd as a proportion of the estimated operating expenses, decimal proportion.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowCritTemp</td>
<td>The animal's lower critical body temperature, degrees Celsius.</td>
</tr>
<tr>
<td>LSCS</td>
<td>The cost of livestock supplies and other expendables for the cow-calf herd, $/cow.</td>
</tr>
<tr>
<td>LSCSS</td>
<td>The cost of livestock supplies and other expendables for the stocker operation, $/head.</td>
</tr>
<tr>
<td>LSUP</td>
<td>The cost of livestock supplies used in the cow-calf operation.</td>
</tr>
<tr>
<td>LSUPS</td>
<td>The total cost of livestock supplies used in the stocker operation.</td>
</tr>
<tr>
<td>mainFDM</td>
<td>The dry matter content of the forage, decimal proportion.</td>
</tr>
<tr>
<td>mainSDM</td>
<td>The dry matter content of the supplemental feed, decimal proportion.</td>
</tr>
<tr>
<td>mainSDMC</td>
<td>The dry matter content of the supplemental feed fed to the breeding herd, decimal proportion.</td>
</tr>
<tr>
<td>MARCHGROWTH</td>
<td>The total amount of forage growth in March, Lbs/acre.</td>
</tr>
<tr>
<td>MaximumNumber</td>
<td>This sets the maximum size of the random number that can be returned.</td>
</tr>
<tr>
<td>MAYGROWTH</td>
<td>The total amount of forage growth in May, Lbs/acre.</td>
</tr>
<tr>
<td>MCADG</td>
<td>A random variable defining an</td>
</tr>
</tbody>
</table>

323
animal's potential average daily gain. It is a normally distributed breed parameter.

MCAGE A random variable that gives the age of a purchased bred heifer. It is normally distributed.

MCARAIN1 A randomly and normally distributed variable describing the precipitation occurring in the Tidewater region of Virginia in the year prior to the start of the simulation.

MCARAIN2 A randomly and normally distributed variable describing the precipitation occurring in the Piedmont region of Virginia in the year prior to the start of the simulation.

MCARAIN3 A randomly and normally distributed variable describing the precipitation occurring in the Mountain region of Virginia in the year prior to the start of the simulation.

MCBWT A random variable giving an animal's potential birth weight. It is a normally distributed breed parameter.

MCCO The unit cost of non-crop machinery used for the cow-calf herd, $/cow.
MCCOS  The unit cost of non-crop machinery used for the stocker herd, $/head.

MCCYC  A random variable describing the length of the estrus cycle. It is a normally distributed variable.

MCDET  A random variable between zero and one (all values equally probable) that will be tested against a constant to determine whether the cow now showing estrus will be detected or not.

MCDIE  A random variable between zero and one (all values equally probable) that will be tested against a constant to determine whether an animal dies at a particular time.

MCDYSF  A random variable between zero and one (all values equally probable) that will be tested against a constant to determine whether dystocia occurs at parturition.

MCGEST  A randomly and normally distributed variable indicating the length of a pregnancy in days. It is a breed parameter.

MCHOW  A randomly and normally distributed variable defining the stage of
pregnancy in days of a purchased bred heifer.

MCMFC A randomly and normally distributed variable indicating a lactating female’s milk fat, percentage.

MCMWT A randomly and normally distributed variable describing an animal's potential mature body weight. It is a breed parameter.

MCPPI A randomly and normally distributed variable indicating the length of the postpartum interval.

MCPREG A random variable between zero and one (all values equally probable) that will be tested against a constant to determine whether female becomes pregnant after being serviced.

MCOAGE A randomly and normally distributed variable indicating the age of a purchased open heifer.

MCRAIN1[13] A randomly and normally distributed variable indicating the precipitation in the current month for the region where the farm is located.

MCRAIN2[13] A randomly and normally distributed variable indicating the precipitation
level in the current month for the region where the farm is located.

**MCRAIN3[13]** A randomly and normally distributed variable indicating the precipitation level in the current month for the region where the farm is located.

**MCSEXN** A random variable between zero and one (all values equally probable) that will be tested against a constant to determine whether fetus will be a male.

**MCSNF** A randomly and normally distributed variable indicating a female’s milk solids percentage.

**MCSTER** A random variable between zero and one (all values equally probable) that will be tested against a constant to determine whether a female becomes chronically unable to conceive or maintain a pregnancy longer than 16 days.

**mcx** The final value of a normally distributed parameter.

**ME** The metabolizable energy of the diet.

**MECO** The total machinery cost for the cow-calf operation.

**MEIntake** The animal's total daily...
metabolizable energy intake, Mcal/day.

**MESO** The total machinery charges for the stocker calf operation.

**MHETADG** The increase in the calf’s pre-weaning potential average daily gain (PADG) that results from maternal heterosis.

**MHETBWT** The increase in the calf’s potential birthweight that results from maternal heterosis.

**MIDN** The identification number of an animal’s mother.

**MONTH** The current month being simulated.

**mother** A temporary variable used in some routines to set or change values in the record structure of a fetus's or calf's dam.

**NBULLS** The total number of breeding bulls maintained in the herd.

**NCCS** The number of culled cows that are available for sale.

**NCTB** The number of live calves born during the year.

**NEG** The net energy of the diet available for gain, Mcal/Kg.

**NEM** The net energy of the diet available
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETIC</td>
<td>The total net income/loss attributed to the cow-calf business.</td>
</tr>
<tr>
<td>NETICS</td>
<td>The total additions to the net income/loss from the stocker calf operation.</td>
</tr>
<tr>
<td>NHFC</td>
<td>The number of heifer calves available for sale at weaning during the simulation year.</td>
</tr>
<tr>
<td>NHFC1</td>
<td>Indicates the number of stocker heifers available for sale during the previous year.</td>
</tr>
<tr>
<td>NHFC2</td>
<td>The number of stocker heifers purchased in the preceding year.</td>
</tr>
<tr>
<td>NIPSH</td>
<td>The number of times that a stocker heifer will be implanted: 1, 2, or 3.</td>
</tr>
<tr>
<td>NIPSS</td>
<td>The number of times that a stocker steer will be implanted: 1, 2, or 3.</td>
</tr>
<tr>
<td>NOC</td>
<td>The number of orphan calves that are available for sale.</td>
</tr>
<tr>
<td>NODET</td>
<td>The probability of an estrus being undetected, i.e., a silent heat. It is a Bernoulli variable.</td>
</tr>
<tr>
<td>NOVEMBERGROWTH</td>
<td>The total amount of forage growth in November, Lbs/acre.</td>
</tr>
<tr>
<td>NSHFC</td>
<td>The number of stocker heifers available</td>
</tr>
</tbody>
</table>
The number of stocker steers available for sale during the simulation year.

NSSTC

The number of steer calves available for sale at weaning during the simulation year.

NSTC

The number of steers calves available for sale.

NSTC1

The number of steer calves in the preceding simulation year.

NSTC2

The number of female calves available to serve as breeding herd replacements.

NumAvailRepla

The number of breeding seasons scheduled during the year, 1 or 2.

NumBreedSeasns

Number of cows on hand at the start of the regular breeding season.

NumCows

The number of breeding cattle of all ages at year end.

NumCowsYend

Number of heifers on hand at the start of the heifer breeding season.

NumHeifers

The number of stocker calves on hand at year end.

NumStkrYEnd

Revenue from the sale of an orphan calf.

OCRV

The total amount of forage growth in October, Lbs/acre.

OCTOBERGROWTH
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>The animal's age in years.</td>
</tr>
<tr>
<td>ONIT</td>
<td>Indicates if cattle will be overnigheted at the auction barn or the Tel-O-Market gathering yard.</td>
</tr>
<tr>
<td>PADG</td>
<td>The animal's potential average daily gain.</td>
</tr>
<tr>
<td>PADGAVG</td>
<td>The potential average daily gain for an animal of a particular genetic composition.</td>
</tr>
<tr>
<td>PADGSD</td>
<td>Indicates the standard deviation of the potential average daily gain for an animal of a particular genetic composition.</td>
</tr>
<tr>
<td>PADMJ</td>
<td>The gross adjustment in the average daily forage yield resulting from erosion, slope, grazing management, weeds, pasture maintenance, and monthly rainfall.</td>
</tr>
<tr>
<td>PARIE</td>
<td>Indicates the parity of a female.</td>
</tr>
<tr>
<td>PAST</td>
<td>Indicates that an animal is on pasture.</td>
</tr>
<tr>
<td>pasture</td>
<td>The number of pastures on the farm.</td>
</tr>
<tr>
<td>PAVAIL</td>
<td>The total amount of forage available for animal consumption in dry matter, Kg.</td>
</tr>
<tr>
<td>PDMI</td>
<td>The total predicted dry matter intake</td>
</tr>
</tbody>
</table>
from forage and supplemental feeds, Kg.

**PDMIF**
The estimated daily dry matter intake from forage for the cow-calf operation.

**PDMIF A**
The estimated daily dry matter intake from forage adjusted for grazing for the cow-calf operation, Kg.

**PDMIS**
The estimated daily dry matter intake from supplemental feeds for the cow-calf operation, Kg.

**PER[13]**
An array storing the genetic composition of a typical cow in the breeding herd of each sire breed listed.

**permPAST**
The total acres of pasture on the farm available for grazing during each month of the simulation year.

**permRPAST**
The total number of acres of rental pasture available for grazing during each month of the simulation year.

**PFOR**
A variable identifying a forage on the input list.

**PFOR1[13]**
Monthly growth rates for FESCUE.

**PFOR2[13]**
Monthly growth rates for a 60:40 combination of FESCUE-LADINO CLOVER.

**PFOR3[13]**
Monthly growth rates for KENTUCKY BLUEGRASS.
PFOR17[13] Monthly growth rates for RYEGRASS-
150N.


PGAVG Indicates the average potential gestation length for an animal of a particular genetic composition.

PGEST The total length of the current pregnancy in days.

PGSD Indicates the standard deviation of the potential gestation length for an animal of a particular genetic composition in days.

PHEF The number of bred heifers purchased as breeding herd replacements during the simulation year.

PMAIN The change in forage yield as a result of pasture maintenance.

PMAN The change in forage yield resulting from the grazing management system in use.

PMNA Pasture maintenance cost charged
against the cow-calf herd, $/acre.

**PMNS**  Pasture maintenance cost charged against the stocker operation, $/acre.

**PMSD**  The standard deviation of the potential mature weight for an animal of a particular genetic composition.

**PMSWT**  The potential slaughter weight of the animal.

**PMSTADG**  The average potential slaughter weight for an animal of a particular genetic composition.

**PMSSD**  The standard deviation of the potential mature weight for an animal of a particular genetic composition.

**PMTAVG**  The average potential mature weight for an animal of a particular genetic composition.

**PMWT**  The animal's potential mature body weight.

**PNUM**  The number of pregnant females.

**POHEF**  The number of nonpregnant heifers purchased as herd replacements during the simulation year.

**PPI**  The female's current postpartum interval.
PRAC  Pasture rental applied to the cow-calf operation, $/acre.
PRAS  Pasture rental applied to the stocker calf operation, $/acre.
PRAX  The total FICA (payroll) taxes charged to the cow-calf operation.
PRAXS The total FICA (payroll) taxes charged to the stocker calf operation.
PRECC  The breed premium for an Angus culled cow.
PRECCC The premium for a black colored culled cow.
PREG  Indicates whether a female is pregnant: 0 = Not pregnant; 1 = Pregnant.
PregAdjust  The price adjustment for pregnant replacements.
PregCow1  The number of pregnant cows during the first third of the regular breeding season for spring calving.
PregCow2  The number of pregnant cows during the second third of the regular breeding season for spring calving.
PregCow3  The number of pregnant cows during the last third of the regular breeding season for spring calving.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PregCow21</td>
<td>The number of pregnant cows during the first third of the regular breeding</td>
</tr>
<tr>
<td></td>
<td>season for fall calving.</td>
</tr>
<tr>
<td>PregCow22</td>
<td>The number of pregnant cows during the second third of the regular breeding</td>
</tr>
<tr>
<td></td>
<td>season for fall calving.</td>
</tr>
<tr>
<td>PregCow23</td>
<td>The number of pregnant cows during the last third of the regular breeding</td>
</tr>
<tr>
<td></td>
<td>season for fall calving.</td>
</tr>
<tr>
<td>PregCullCow</td>
<td>The annual number of pregnant cows</td>
</tr>
<tr>
<td></td>
<td>culled for age restrictions.</td>
</tr>
<tr>
<td>PregHef1</td>
<td>The number of pregnant heifers during the first third of the heifer breeding</td>
</tr>
<tr>
<td></td>
<td>season for spring calving.</td>
</tr>
<tr>
<td>PregHef2</td>
<td>The number of pregnant heifers during the second third of the heifer breeding</td>
</tr>
<tr>
<td></td>
<td>season for spring calving.</td>
</tr>
<tr>
<td>PregHef3</td>
<td>The number of pregnant heifers during the last third of the heifer breeding</td>
</tr>
<tr>
<td></td>
<td>season for spring calving.</td>
</tr>
<tr>
<td>PregHef21</td>
<td>The number of pregnant heifers during the first third of the heifer breeding</td>
</tr>
<tr>
<td></td>
<td>season for fall calving.</td>
</tr>
<tr>
<td>PregHef22</td>
<td>The number of pregnant heifers during the second third of the heifer breeding</td>
</tr>
<tr>
<td></td>
<td>season for fall calving.</td>
</tr>
</tbody>
</table>
PregHef23 The number of pregnant heifers during the last third of the heifer breeding season for fall calving.

PREH The breed premium for an Angus heifer.

PREHC The color premium for a black colored heifer.

PREP The change in forage yield resulting from the monthly precipitation.

PRES The breed premium for an Angus steer.

PRESC The color premium for a black colored steer.

PRH[9] An array that contains the price adjustments for heifers from 300 - 800 pounds, \(i = 4 = 300 - 400 \text{ lbs.};\ \ i = 5 = 400 - 500 \text{ lbs.};\ \ i = 6 = 500 - 600 \text{ lbs.};\ \ i = 7 = 600 - 700 \text{ lbs.};\ \ i = 8 = 700 - 800 \text{ lbs.}\).

PRH[11] An array that contains the price adjustments for stocker heifers from 500 - 1000 pounds, \(i = 6 = 500 - 600 \text{ lbs.};\ \ i = 7 = 600 - 700 \text{ lbs.};\ \ i = 8 = 700 - 800 \text{ lbs.};\ \ i = 9 = 800 - 900 \text{ lbs.};\ \ i = 10 = 900 - 1000 \text{ lbs.}\).

PRIC The price adjustment for a commercial grade cull cow.
PRIS  The specified base price for a 600 – 700 lb. steer calf.

PRIU  The price adjustment for a utility grade cull cow.

PRS[9]  An array that lists the price adjustments for steer calves from 300 – 800 pounds, (i = 4 = 300 – 400 lbs.; i = 5 = 400 – 500 lbs.; i = 6 = 500 – 600 lbs.; i = 7 = 600 – 700 lbs.; i = 8 = 700 – 800 lbs.


PRSA1  The estimated average price of a 600 – 700 lb. Virginia steer in Year 1.

PRSA2  The estimated average price of a 600 – 700 lb. Virginia steer in Year 2.

PRSA3  The estimated average price of a 600 – 700 lb. Virginia steer in Year 3.

PRS[13]  An array indicating the estimated average monthly price per hundred weight of a 600 – 700 lb. steer during simulation Year 1.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRS2[13]</td>
<td>An array indicating the estimated average monthly price per hundred weight of a 600 - 700 lb. steer during simulation Year 2.</td>
</tr>
<tr>
<td>PRS3[13]</td>
<td>An array indicating the estimated average monthly price per hundred weight of a 600 - 700 lb. steer during simulation Year 3.</td>
</tr>
<tr>
<td>PSBWMK</td>
<td>The potential mature shrunk body weight of the animal in kilograms.</td>
</tr>
<tr>
<td>PTEST</td>
<td>The number of days after the last day of the breeding season that the breeding herd is pregnancy tested.</td>
</tr>
<tr>
<td>PUBER</td>
<td>The date of the pubertal estrus.</td>
</tr>
<tr>
<td>Puberty</td>
<td>Indicates whether the breeding female has reached puberty: 0 = No; 1 = Yes.</td>
</tr>
<tr>
<td>PurchaseCost</td>
<td>The cost of each individual cow or calf purchased.</td>
</tr>
<tr>
<td>PURE</td>
<td>Indicates whether an animal is purebred or crossbred: 0 = a purebred animal; 1 = a crossbred animal.</td>
</tr>
<tr>
<td>PW</td>
<td>The peak week of milk production if the animal is lactating.</td>
</tr>
<tr>
<td>q</td>
<td>A local counter.</td>
</tr>
<tr>
<td>R</td>
<td>A local variable</td>
</tr>
<tr>
<td>RAINAPRIL</td>
<td>The estimated amount of rainfall.</td>
</tr>
</tbody>
</table>

340
occurring in April.

RAINAUGUST The estimated amount of rainfall occurring in August.


RAINDECEMBER The estimated amount of rainfall occurring in December.


<table>
<thead>
<tr>
<th>Monthly Rainfall</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAINFEBRUARY</td>
<td>The estimated amount of rainfall occurring in February.</td>
</tr>
<tr>
<td>RAINJANUARY</td>
<td>The estimated amount of rainfall occurring in January.</td>
</tr>
<tr>
<td>RAINJULY</td>
<td>The estimated amount of rainfall occurring in July.</td>
</tr>
<tr>
<td>RAINJUNE</td>
<td>The estimated amount of rainfall occurring in June.</td>
</tr>
<tr>
<td>RAINMARCH</td>
<td>The estimated amount of rainfall occurring in March.</td>
</tr>
<tr>
<td>RAINMAY</td>
<td>The estimated amount of rainfall occurring in May.</td>
</tr>
<tr>
<td>RAINNOVEMBER</td>
<td>The estimated amount of rainfall occurring in November.</td>
</tr>
<tr>
<td>RAINOCTOBER</td>
<td>The estimated amount of rainfall occurring in October.</td>
</tr>
<tr>
<td>RAINSEPTEMBER</td>
<td>The estimated amount of rainfall occurring in September.</td>
</tr>
</tbody>
</table>

randomnumber: A local variable that is assigned the value of the last random number generated.

RE: The amount of retained energy, Mcal/day.

RegTemp[4][2][13]: The average temperature and standard deviation of the temperature during each month in the Tidewater Region of
Virginia.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPLA</td>
<td>Indicates whether the female calf is a breeding herd replacement: 0 = not a breeding herd replacement; 1 = a breeding herd replacement.</td>
</tr>
<tr>
<td>REPLF</td>
<td>The number of breeding herd replacement females purchased during the simulation year.</td>
</tr>
<tr>
<td>REPLR</td>
<td>The number of breeding herd replacement females required.</td>
</tr>
<tr>
<td>REPNUM</td>
<td>The total number of replacement heifers in the herd at the start of the simulation.</td>
</tr>
<tr>
<td>RESP</td>
<td>The user's responsibility for pasture maintenance for acreage assigned to the cow-calf operation, percentage.</td>
</tr>
<tr>
<td>RESPS</td>
<td>The user's responsibility for pasture maintenance for acreage assigned to the stocker operation, percentage.</td>
</tr>
<tr>
<td>Rpasture</td>
<td>The number of pastures rented for the stocker operation.</td>
</tr>
<tr>
<td>SALC</td>
<td>Indicates the marketing method used to sell culled cows: 0 = direct marketing; 1 = auction barns.</td>
</tr>
<tr>
<td>SALH</td>
<td>Indicates the marketing method used to sell heifer calves: 0 = direct</td>
</tr>
</tbody>
</table>
marketing; 1 = auction barns; 2 = Tel-O-Market auctions.

SALO Indicates the marketing method used to sell orphan calves: 0 = direct marketing; 1 = auction barns.

SALS Indicates the marketing method used to sell steer calves: 0 = direct marketing; 1 = auction barns; 2 = Tel-O-Market auctions.

SALSH Indicates the marketing method used to sell stocker heifers: 0 = direct marketing; 1 = auction barns; 2 = Tel-O-Market auctions.

SALSS Indicates the marketing method used to sell stocker steers: 0 = direct marketing; 1 = auction barns; 2 = Tel-O-Market auctions.

SALT The annual unit consumption of salt and minerals for the cow-calf herd, hundredweight/cow.

SAME The metabolizable energy of the supplemental feed, Mcal/Kg.

SANEG The available net energy for gain from the supplemental feeds, Mcal/Kg.

SANEM The available net energy for maintenance from the feed
supplement, Mcal/Kg.

**SBWTG** The daily gain in shrunk body weight of the animal in kilograms.

**SBWTK** The current shrunk body weight of the animal in kilograms.

**SCALF** The total number of spring calves weaned.

**SCC** The dollar premium from the direct sale of culled cows.

**SDM[21]** An array storing the dry matter content of the supplemental feeds ingredients, decimal proportion.

**SEAOP** The number of breeding seasons that a female has not conceived.

**SEAOP1** Indicates whether the female failed to conceive during the first breeding season when both spring and fall breeding seasons are used.

**SEAOP2** Indicates if the female failed to conceive during the second annual breeding season when both spring and fall breeding seasons are used.

**SEAS** The length of the regular breeding season in days.

**SEASON** Indicates the breeding season used:
1 = Spring calving; 2 = Fall calving.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEAS2</td>
<td>The length of the fall breeding season for the current year.</td>
</tr>
<tr>
<td>SELEC</td>
<td>A counter that records the number of female calves selected to be breeding herd replacements.</td>
</tr>
<tr>
<td>SelectWIHerd</td>
<td>Indicates if replacements are to be selected from within the herd before purchasing open heifers: 0 = NO; 1 = YES.</td>
</tr>
<tr>
<td>SELL</td>
<td>The average age of the stocker calves at the time of sale.</td>
</tr>
<tr>
<td>SellCalf</td>
<td>Indicates that an orphan calf is to be sold immediately and not transferred to the stocker herd: 0 = do not sell calf; 1 = sell calf.</td>
</tr>
<tr>
<td>SELLHAY</td>
<td>The sales value of the hay in dollars/ton.</td>
</tr>
<tr>
<td>SENE</td>
<td>The daily pasture senescence, decimal proportion.</td>
</tr>
<tr>
<td>SEPTEMBERGROWTH</td>
<td>The total amount of forage growth in September, Lbs/acre.</td>
</tr>
<tr>
<td>SERVI</td>
<td>The number of fertile estrous periods during the breeding season required for a cow to become pregnant.</td>
</tr>
<tr>
<td>Servi</td>
<td>The number of services per conception across the female breeding herd, i.e.,</td>
</tr>
</tbody>
</table>
one service is counted for each estrous period during the breeding season.

Servil  
The cumulative number of services per conception when breeding for fall calving.

SEX  
The sex of the animal: 0 = female prior to weaning; 2 = castrated male prior to weaning; 3 = a breeding herd female; 4 = a stocker female after weaning; 5 = a stocker male after weaning.

SEXM  
The probability of a calf being a male. It is a Bernoulli variable.

SHFC  
The dollar premium for the direct sales of heifers.

SHFE  
The dollar premium for the Tel-O-Market auction sales of heifers.

SHFRV  
The revenue obtained from the sale of a stocker heifer.

SHKC  
The proportional shrinkage on any culled cow sold.

SHKCP  
The pencil allowance shrinkage on the direct sale of culled cows, decimal proportion.

SHKH  
The shrinkage on any heifer, decimal proportion.

SHKHP  
The pencil allowance shrinkage on the
direct sale of heifers, decimal proportion.

SHKOC The shrinkage on any orphan calf, decimal proportion.

SHKOP The pencil allowance shrinkage on the direct sale of orphan calves, decimal proportion.

SHKS The shrinkage on any steer calf, decimal proportion.

SHKSH The shrinkage on any stocker heifer, decimal proportion.

SHKSHP The pencil allowance shrinkage on the direct sale of stocker heifers, decimal proportion.

SHKSP The pencil allowance shrinkage on the direct sale of steer calves, decimal proportion.

SHKSS The shrinkage on any stocker steer sold, decimal proportion.

SHKSSP The pencil allowance shrinkage on the direct sale of steer calves, decimal proportion.

simulationDATE The current day of the simulation.

SIREBREED A numerical code identifying the breed of a purebred bull or the sire breed of a crossbred cow: 1 = Angus;
2 = Hereford; 3 = Polled Hereford;
4 = Shorthorn; 5 = Charolais;
6 = Chianina; 7 = Gelbvieh;
8 = Limousin; 9 = Maine Anjou;
10 = Pinzgauer; 11 = Simmental;
12 = Tarentaise.

**SIZE**
The number of pasture acres dedicated to the cow-calf operation.

**SIZES**
The number of pasture acres rented for the stocker calf operation.

**SLOE**
The average amount of slope in the pasture, percentage.

**SLOP**
The reduction in grazing capacity that results from the average slope of the pasture.

**SMCS**
The price of salt and minerals, \$/hundredweight.

**SMCSS**
The price of salt and minerals purchased for the stocker operation, \$/hundredweight.

**SME[21]**
An array storing the metabolizable energy of the supplemental feed ingredients, Mcal/kg.

**SMFC**
The total cost of salt and minerals fed in the cow-calf operation.

**SMFS**
The total cost of salt and minerals
fed to stocker calves.

SNEG[21]  An array storing the net energy for gain of the supplemental feed ingredients, Mcal/kg.

SNEM[21]  An array storing the net energy for maintenance of the supplemental feed ingredients, Mcal/kg.

SOC        The dollar value premium from the direct sale of orphan calves.

SOIL       The reduction in forage yield resulting from pasture erosion.

SOPT0      A variable storing the name of an optional feed ingredient.

SOPT1      A variable storing the name of an optional feed ingredient.

SOPT2      A variable storing the name of an optional feed ingredient.

SOPT3      A variable storing the name of an optional feed ingredient.

SOPT4      A variable storing the name of an optional feed ingredient.

SOPT5      A variable storing the name of an optional feed ingredient.

SOUR       Indicates if the animal has been bred on the farm or purchased: 0 = Bred on the farm; 1 = Purchased. It is used to
calculate the cost of replacements purchased during a simulation year.

SPA[13] An array indicating the monthly price adjustments to the estimated average annual price per hundredweight of a 600 - 700 lb. feeder steer during years 1 - 3 of the simulation.

SPDMI The total predicted dry matter intake from the forage plus any supplemental feeds for the stocker calves, Kg.

SPDMIF The estimated daily dry matter intake from forage for stocker calves, Kg.

SPDMIFA The estimated daily dry matter intake from forage adjusted for grazing for the stocker calves, Kg.

SPDMIS The estimated daily dry matter intake from supplemental feeds fed to stocker calves, Kg.

SPRH[11] An array defining the price adjustments for heifers from 300 - 1000 pounds, (i = 4 = 300 - 400 lbs.; i = 5 = 400 - 500 lbs.; i = 6 = 500 - 600 lbs.; i = 7 = 600 - 700 lbs.; i = 8 = 700 - 800 lbs.; i = 9 = 800 - 900 lbs.; i = 10 = 900 - 1000 lbs.

adjustments for steers from 300 - 1000 lbs., (i = 4 = 300 - 400 lbs.; i = 5 = 400 - 500 lbs.; i = 6 = 500 - 600 lbs.; i = 7 = 600 - 700 lbs.; i = 8 = 700 - 800 lbs; i = 9 = 800 - 900 lbs.; i = 10 = 900 - 1000 lbs.

SPRSA1 The estimated average price of a 700 - 800 lb. Virginia yearling steer in Year # 1 as specified by the user.

SPRSA2 The estimated average price of a 700 - 800 lb. Virginia yearling steer in Year # 2 as specified by the user.

SPRSA3 The estimated average price of a 700 - 800 lb. Virginia yearling steer in Year # 3 as specified by the user.

SREG The region of Virginia in which the farm is located: 1 = Tidewater Region; 2 = Piedmont region; 3 = Mountain Region.

SRWK An NRC reference weight that represents the current cross section of breed and body sizes in the U. S., 478 kg.

SSC The dollar premium for the direct sale of steer calves.

SSE The dollar premium for Tel-O-Market auction sales of steer calves.
SSELLDATE The day on which stocker calves born
during the spring calving season are sold.

SSHFC The dollar premium for the direct sale of stocker heifers.

SSHFE The dollar premium for the Tel-O-Market auction sales of stocker heifers.

SSSC The dollar premium for the direct sale of stocker steers.

SSSE The dollar premium for Tel-O-Market auction sales of stocker steers.

SSTRV The marketing revenue from each stocker steer sold.

STDN[21] An array storing the TDN for each supplemental feed ingredient, percentage.

Sterile Indicates the number of females in the Breeding herd that are estimated to be chronically unable to maintain a pregnancy longer than 16 days.

STFeedCost The annual cost of supplemental feeds consumed by the stocker calf herd, dollars.

STRV The marketing revenue from each steer calf sold.
SU[21] An array containing the amount of each ingredient contained in the feed supplement, percentage.

SUPP The proportion of the diet not supplied by grazing, decimal proportion.

SUPPL Indicates whether supplemental feeds are fed: 0 = No; 1 = Yes.

SurArea The animal's surface area, m².

TAIC The total cost of artificially inseminating the cow-calf herd.

TAPIS The total additional projected income/loss from the stocker calf operation.

TAPG The total annual pasture growth in Kg DM/Ha.

TBDCC The total breed and color premium received for culled cows.

TBDH The total breed or color premium received for heifers.

TBDH1 The total breed premium/discount for heifers based on breed or color in YEAR 0 of the warm-up period.

TBDOC The total breed or color premium received for orphan calves.

TBDS The total breed or color premium received for steer calves.
TBDS1  The total breed premium/discount for steer calves based on breed or color in YEAR 0 of the warm-up period.

TBDSH  The total breed and color premium received for stocker heifers.

TBDSS  The total breed and color premium received for stocker heifers.

TBRED[13]  The total number of cows in the original breeding herd of any particular breed.

TBWT  The total birth weight of calves born during the simulation year.

TBWTH  The total birth weight of heifers born during the simulation year.

TBWTS  The total birth weight of steers born during the simulation year.

Tc  The current average monthly temperature, degrees Celsius.

TCCR  The marketing revenue received from each culled cow sold.

TCCS  The cost of transporting one culled cow to market, $/head.

TCDAT  The total of all calving dates.

TCPDMIS  The total predicted dry matter intake from supplemental feeds for the cows of the cow-calf operation, Kg.
TCSC  The cost of transporting one calf to market, $/head.

TCSCS The cost of transporting a stocker calf to market, $/head.

TDNP  The total digestible nutrient content of the pasture forage.

TDSPC The total dollar premium received for the direct sale of culled cows.

TDSPH The total sales premium received for the direct marketing of heifers.

TDSPH1 The total sales premium received for the direct marketing of heifers in YEAR 0 of the warm-up period.

TDSPOC The total sales premiums received for the direct marketing of orphans.

TDSPS The total sales premium received for the direct marketing of steer calves.

TDSPS1 The total sales premium received for the direct marketing of steer calves in YEAR 0 of the warm-up period.

TDSPSH The total dollar premium for the direct marketing of stocker heifers.

TDSPSS The total dollar premium for the direct marketing of stocker steers.

TECCE The total expenses for the cow-calf operation.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TELO</td>
<td>The telephone charges for each weanling and stocker calf sold through an electronic auction.</td>
</tr>
<tr>
<td>temp</td>
<td>Used in sorting the linked list.</td>
</tr>
<tr>
<td>TempCDWTK</td>
<td>Indicates the animal’s current live body weight, Kg.</td>
</tr>
<tr>
<td>tempcow</td>
<td>A temporary animal used to either place new animals on the linked list or make changes to the linked list.</td>
</tr>
<tr>
<td>TempEQS BW</td>
<td>Indicates the animal’s equivalent shrunk body weight, Kg.</td>
</tr>
<tr>
<td>tempmother</td>
<td>A temporary animal used to make changes to the linked list.</td>
</tr>
<tr>
<td>TempSBWTK</td>
<td>Indicates the animal’s shrunk body weight, Kg.</td>
</tr>
<tr>
<td>TERR</td>
<td>The general terrain of the farm: 1 = level; 2 = hilly.</td>
</tr>
<tr>
<td>TESPH</td>
<td>The total sales premium received for the Tel-O-Market auction marketing of heifers.</td>
</tr>
<tr>
<td>TESPH1</td>
<td>The total sales premium received for the Tel-O-Market auction marketing of heifers in YEAR 0 of the warm-up period.</td>
</tr>
<tr>
<td>TESPS</td>
<td>The total sales premium received for the Tel-O-Market auction marketing of</td>
</tr>
</tbody>
</table>
steer calves.

TESPS1 The total sales premium received for the Tel-O-Market auction marketing of steer calves in YEAR 0 of the warm-up period.

TESPSH The total dollar premium for the Tel-O-Market auctions of stocker steers.

TESPSS The total dollar premium for the Tel-O-Market auctions of stocker heifers.

TEST Indicates whether the herd will be pregnancy tested after the breeding season is over: 0 = YES; 1 = NO.

TESTE The total additional expenses attributed to the stocker calf operation.

TFECA The total fees charged for selling all cattle except those marketed directly.

TFECA1 The total fees charged for selling all cattle except those marketed directly in YEAR 0 of the warm-up period.

TFECAC The total fees charged for selling all culled cows except those sold by directing marketing.

TFECAS The total fees charged for selling
all stocker cattle.

**TFOR**
This indicates the total number of forages found in the grazing unit.

**TFR**
The total forage composition of the pasture, decimal proportion. It is used to ensure that the user does not input more than a forage composition equal to 1.0.

**TFWEANAGE**
The total weaning age of all fall calving season calves.

**TGRO**
The total growth of the combined forage species in pounds per acre.

**THFR**
The total sales revenues received for all heifers sold.

**THFR1**
The total sales revenues received for all heifers sold in YEAR 0 of the warm-up period.

**time_t**
The current computer clock time.

**TNCOH**
The total number of calves on hand at year end.

**TNCS**
The total number of weanling calves sold.

**TNCW**
The total number of calves weaned during the year.

**TNSCOH**
The total number of stocker calves on hand at year end.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNCS</td>
<td>The total number of stocker cattle sold.</td>
</tr>
<tr>
<td>TOEX</td>
<td>The total operating expenses for the cow-calf operation.</td>
</tr>
<tr>
<td>TOEXS</td>
<td>The total additional operating expenses for the stocker calf operation.</td>
</tr>
<tr>
<td>TOCR</td>
<td>The total sales revenues received for all orphan calves sold.</td>
</tr>
<tr>
<td>TotalPurchaseCost</td>
<td>The total cost of cattle purchased.</td>
</tr>
<tr>
<td>TOTALAGE</td>
<td>The combined ages of the breeding herd at year end.</td>
</tr>
<tr>
<td>TotAnnSen</td>
<td>The total annual forage senescence in lbs.</td>
</tr>
<tr>
<td>TotGestLen</td>
<td>Indicates the total combined length of gestation for pregnant cows within the herd.</td>
</tr>
<tr>
<td>TotWeanWtH</td>
<td>The total weaning weight of stocker heifers sold during the year.</td>
</tr>
<tr>
<td>TotWeanWtS</td>
<td>The total weaning weight of stocker steers sold during the year.</td>
</tr>
<tr>
<td>Tp</td>
<td>The previous month's average temperature, degrees Celsius.</td>
</tr>
<tr>
<td>TPAS[7]</td>
<td>An array containing the total number of acres in each pasture used as permanent pasture.</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>TPCA</td>
<td>The total cost of hauling calves to market.</td>
</tr>
<tr>
<td>TPCC</td>
<td>The total cost of hauling culled cows to market.</td>
</tr>
<tr>
<td>TPCSC</td>
<td>The total cost of hauling stocker cattle to market.</td>
</tr>
<tr>
<td>TPDMIF</td>
<td>The total estimated dry matter intake from forage used by the cow-calf operation, Kg.</td>
</tr>
<tr>
<td>TPDMIFA</td>
<td>The total estimated daily dry matter intake used by the cow-calf operation adjusted for the grazing situation, Kg.</td>
</tr>
<tr>
<td>TPDMIS</td>
<td>The total predicted dry matter intake from supplemental feeds for the cow-calf operation, Kg.</td>
</tr>
<tr>
<td>TPIC</td>
<td>Total projected income/loss in the cow-calf operation.</td>
</tr>
<tr>
<td>TPMN</td>
<td>Total pasture maintenance expenses for the cow-calf operation.</td>
</tr>
<tr>
<td>TPMNS</td>
<td>Total pasture maintenance expenses for any additional acreage rented for the stocker calf operation.</td>
</tr>
<tr>
<td>TPRD</td>
<td>Total pasture rental expenses for the cow-calf operation.</td>
</tr>
<tr>
<td>TPRDS</td>
<td>Total pasture rental expenses for any</td>
</tr>
</tbody>
</table>
additional acreage rented for the stocker calf operation.

TPURWT The total purchase weight of open replacement heifers.

TPWBH The total purchase weight of bred replacement heifers.

TRPAS[7] An array containing the total number of acres in each stocker rental pasture used as permanent pasture.

TSHFR The total revenues received for stocker heifers sold.

TSHKC The total shrinkage of culled cows sold.

TSHKH The total shrinkage of heifers sold.

TSHKO The total shrinkage of orphans sold.

TSHKS The total shrinkage of steers sold.

TSHKSH The total shrinkage of stocker heifers sold.

TSHKSS The total shrinkage of stocker steers sold.

TSPDMIF The total estimated dry matter intake from forage for the stocker calf operation, Kg.

TSPDMIFA The total estimated daily dry matter intake adjusted for the grazing situation for the stocker calf
operation, Kg.

TSPDMIS The total estimated dry matter intake from supplemental feeds for the stocker calf operation, Kg.

TSSTR The total revenues received for stocker steers sold.

TSTOP Indicates the number of cows that tested nonpregnant during the year.

TSTR The total sales revenues received for steer calves sold.

TSTR1 The total sales revenues received for all steer calves sold in YEAR 0 of the warm-up period.

TSU The total composition of ingredients contained in the supplemental feed. It ensures that the user cannot input more supplemental feeds than 100%, percentage.

TSWEANAGE The total weaning age of spring calving season calves.

TTCCR The total annual culled cow revenue.

TTGRO Indicates the total pasture growth available for grazing before deducting senescence.

TTHFR The total annual heifer revenue.

TTHFR0 The total annual heifer revenue in
YEAR 0 of the warm-up period.

TTHFR1 Indicates the total annual heifer revenue for the current year.

TTHFR2 Indicates the total revenues received from the sale of stocker heifers during the current year.

TTOCR The total annual orphan calf revenue.

TTSHFR The total stocker heifer revenue.

TTSSTR The total stocker steer revenue.

TTSTR The total annual steer revenue.

TTSTR0 The total annual steer revenue in YEAR 0 of the warm-up period.

TTSTR1 Indicates the total annual steer revenue for the current year.

TTSTR2 Indicates the total cost of the stocker steers sold during the current year.

TWCC Indicates the total weight of culled cows sold.

TWCS The total weight of all weanling calves sold.

TWHS Indicates the total weight of heifer calves sold.

TWHS1 Indicates the total weight of heifer calves sold during the current year.

TWHS2 Indicates the total beginning weight of stocker heifer calves sold in the
current year.

**TWOS** Indicates the total weight of orphan calves sold.

**TWSCS** The total weight of the stocker cattle sold, lbs.

**TWSHS** Indicates the total weight of stocker heifers sold.

**TWSS** Indicates the total weight of steer calves sold.

**TWSS1** Indicates the total weight of steer calves sold during the current year.

**TWSS2** Indicates the total beginning weight of stocker steers sold in the current year.

**TWSSS** Indicates the total weight of stocker steers sold.

**UADG** The user-provided post-weaning mean ADG for the stocker herd.

**UADG1** The user-provided post-weaning mean ADG for the stocker herd from January 1 through March 31.

**UADG2** The user-provided post-weaning mean ADG for the stocker herd from April 1 through June 30.

**UADG3** The user-provided post-weaning mean ADG for the stocker herd from July 1
through September 30.

UADG4 The user-provided post-weaning mean ADG for the stocker herd from September 1 through December 31.

UNPREG Indicates whether the female is chronically unable to conceive or maintain a pregnancy longer than 16 days: 0 = the female is able to conceive or maintain a pregnancy longer than 16 days; 1 = the female is chronically unable to conceive or maintain a pregnancy longer than 16 days.

UNPREGC The average proportion of females with a parity greater than zero that become chronically unable to conceive or maintain a pregnancy longer than 16 days after suffering dystocia. It is a Bernoulli variable.

UNPREGH The average proportion of heifers in the herd with a parity of zero that are chronically unable to conceive or maintain a pregnancy longer than 16 days after suffering dystocia.

UTIL The total cost of utilities for the cow-calf operation.

UTILS The total cost of utilities for the
stocker calf operation.

UTLC The unit cost of utilities used for the cow-calf herd, $/cow.

UTLCH The unit cost of utilities used for the stocker herd, $/head.

VACA The Virginia Cattleman's Association commission fee for each calf sold through its Tel-O-Market auction.

VMCS The cost of veterinary services and medicine for the cow-calf herd, $/cow.

VMCSS The cost of veterinary services and medicine for the stocker operation, $/head.

VMDC The total cost of veterinary services and medicine for the cow-calf operation.

VMDCS The total cost of veterinary services and medicine for the stocker calf operation.

WAGE The gross labor cost, $/hour.

WEAN The age of a calf at weaning.

WEANDATE The date that the calf was weaned from its mother.

WEANFDATE The Julian calendar day on which the fall calves are weaned.
weanfday The day that fall calves are weaned.
WRATE The annual weaning rate, percentage.
WEANSDATE The Julian calendar day on which the spring calves are weaned.
weansday The day that spring calves are weaned.
weanfmonth The month in which fall calves are weaned.
weansmonth The month in which spring calves are weaned.
WEDS The severity of weed infestation on the pasture: 1 = severe; 2 = moderate; 3 = light; 4 = non-existent.
WEED The reduction in forage yield resulting from weed infestation in the pasture.
WIND The average annual wind speed in Virginia, mph.
WLAND[7] An array containing the number of acres of ungrazable woodlands in each pasture.
WNWT The total live weight of the calves weaned during the year.
woodLANDS The total acres of ungrazable woodlands on the farm.
woodRLANDS The total acres of ungrazable woodlands on the rental pastures.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRATE</td>
<td>The annual weaning rate, percentage.</td>
</tr>
<tr>
<td>WRLAND[7]</td>
<td>An array containing the number of acres of ungrazable woodlands in each rental pasture.</td>
</tr>
<tr>
<td>YDCA</td>
<td>The total yardage charge for calves sold through either auction markets or Tel-O-Market gathering pens.</td>
</tr>
<tr>
<td>YDCA1</td>
<td>The total yardage charge on all calves sold through either auction markets or Tel-O-Market gathering pens in YEAR 0 of the warm-up period.</td>
</tr>
<tr>
<td>YDCAC</td>
<td>The total yardage charge for culled cows sold through auction markets.</td>
</tr>
<tr>
<td>YDCAS</td>
<td>The total yardage charge for stocker calves sold through either auction markets or Tel-O-Market gathering pens.</td>
</tr>
<tr>
<td>YFAIL</td>
<td>Used to cull cows by indicating if the animal failed to calf during the previous year.</td>
</tr>
<tr>
<td>YEAR</td>
<td>The current simulation year.</td>
</tr>
<tr>
<td>YEN</td>
<td>The total daily energy provided by milk at the current stage of lactation, Mcal/Kg.</td>
</tr>
<tr>
<td>YN</td>
<td>The daily milk yield, Kg/day.</td>
</tr>
<tr>
<td>YRCD</td>
<td>The total yardage fees that an auction barn charges for each animal</td>
</tr>
</tbody>
</table>
sold through its facilities as a
decimal proportion of sales revenue.

YRD The yardage charged for each calf.

YRDC The yardage charged for each culled
cow.

YRDS The yardage charged for each stocker
calf.

z A local counter variable.

Z A local variable used in the function
GetStochasticNumber.

z1 The first uniformly distributed
random number generated in the function
GetStochasticNumber.

z2 The second uniformly distributed
random number generated in the function
GetStochasticNumber.
Appendix C
The Virginia Beef Cattle Simulation Model
Programming Code
Source Code Files
#include "stdafx.h"
#include "HerdSim.h"
#include "HerdSimDlg.h"
#include <fstream.h>
#include <iostream.h>
#include <time.h>
#include <stdlib.h>
#include <process.h>
#include <math.h>
#include "cow.h"
#include "market.h"
#include "nutrit.h"
#include "forage.h"
#include "main.h"
#include "tools.h"
#ifndef __AFXWIN_H__
#error include 'stdafx.h' before including this file for PCH
#endif
#include "resource.h"
#endif _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
cstatic char THIS_FILE[] = __FILE__;
#endif
//Uncomment this for the herd debug file "cowtest.txt"
//WARNING: this will cause the simulation to decrease in
speed by a factor ~1000
//#define ANIMAL_TRACE
const int CNTR = 50000;
int currentDATE;
int simulationDAY;
int YEAR;
int DAY;
int MONTH;
Main::Main(){
}
void Main::Start(){
Cow *userHerd = new Cow;
Forage *userForage = new Forage;
Market *userMarket = new Market;
//Create the nutrition
Nutrition *userNutrit = new Nutrition;
int lastDAY;
//As we go through the herd
AnimalRecord *currentCow;
AnimalRecord *backupNextCow;
CProgress *cp;
int marketOutput = -1;
time_t t;
int response=9990;
CMainMenu* mainmenu;
srand((unsigned) time(&t));
GetRandom();
ofstream out;
mainmenu = new CMainMenu();
bool stopDay;
#ifdef ANIMAL_TRACE
ofstream test;
test.open("cowtest.txt", ios::out, filebuf::sh_read);
#endif
currentDATE=1;
simulationDAY=1;
TRACE0("Main Menu called\n");
while (response != IDC_QUITSIM){
    response = mainmenu->DoModal();
    if (response == IDC_COW){
        userHerd->Init();
    }
    else if (response == IDC_MARKET){
        userMarket->Init();
    }
    else if (response == IDC_FORAGE){
        userForage->Init();
    }
    else if (response == IDC_NUTRIT){
        userNutrit->Init();
    }
    else if (response == IDC_QUITSIM){
        if (AfxMessageBox("Warning: All Simulation Data will be
Deleted", IDOK) == IDOK){
            response = response;
        }
        else{
            response = -1;
        }
    }
    while (response == IDC_STARTSIM){
        response=-1;
        TRACE0("Going to create herd\n");
        userHerd->CreateHerd();
        cp = new CProgress();
        cp->Create(IDD_PROGRESS, NULL);
        cp->ShowWindow(SW_SHOW);
        cp->onStartUp(11);
        for (YEAR=-7; YEAR < 4; YEAR++){
            TRACE1("YEAR: \n", YEAR);
            cp->onStepFunction();
            currentDATE=1;
            for (MONTH=1; MONTH <= 12; MONTH++){
                DAY=1;
                if ((YEAR>=0) && (MONTH==1)) { //Change Back to
(YEAR>=0) only
ifdef ANIMAL_TRACE
userHerd->DumpList(test);
#endif

switch(MONTH){
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12: lastDAY=31; break;
case 4:
case 6:
case 9:
case 11: lastDAY=30; break;
case 2: lastDAY=28; break;
}

while (DAY <= lastDAY){
userForage->MainForage();
userForage->Temperature();
currentCow=userHerd->GetHeadAnimal();
currentCow=currentCow->next;
if (currentCow == NULL){
CDieHerd* chd = new CDieHerd;
int resp;
CString* newCaption = new CString;
newCaption->Format("%f percent of the time.", (pow(.02,
userHerd->GetOriginalNumOfCows()) * 100));
chd->m_prob = *newCaption;
resp = chd->DoModal();
if (resp == IDOK){
YEAR = 20;
DAY = 500;
MONTH = 13;
response = IDC_STARTSIM;
}
else{
YEAR = 20;
DAY = 500;
MONTH = 13;
}
}
}

while(currentCow != NULL){
backupNextCow = currentCow->next;
if (currentCow->AGE > 0){
if (currentCow->DIE!=1){
userHerd->ShrinkAnimal(currentCow);
userHerd->Dystocia(currentCow);
userHerd->Sterility(currentCow);
userHerd->Parturition(currentCow);
userNutrit->MainNutrition(currentCow);
userNutrit->DWGAIN(currentCow);
userNutrit->FeedUsage();
userHerd->Puberty(currentCow);
userHerd->PregnancyTest(currentCow);
userHerd->CullCows(currentCow);
userHerd->Lactation(currentCow);
userHerd->CalfMilk(currentCow);
userHerd->BreedFemales(currentCow);
userHerd->Abort(currentCow);
userHerd->Mortality(currentCow);
userHerd->SBBreeders(currentCow);
}
if (backupNextCow != currentCow->next){
backupNextCow = currentCow->next;
}
if (userHerd->DeadCattle(currentCow) != 1){
userHerd->Wean(currentCow);
stopDay=true;
userMarket->PurchaseLivestock(currentCow);
userMarket->SellWeanlings(currentCow);
userHerd->ChangeSex(currentCow);
userHerd->ReproStats();
marketOutput = userMarket->SellOrphans(currentCow);
if (marketOutput != -1){
if (marketOutput == currentCow->ID){
userHerd->DeleteAnimal(marketOutput);
currentCow = backupNextCow;
stopDay=true;
}
else{
userHerd->DeleteAnimal(marketOutput);
stopDay=false;
}
}
if (stopDay == false){
marketOutput = userMarket->SellCullCows(currentCow);
if (marketOutput != -1){
if (marketOutput == currentCow->ID){
userHerd->DeleteAnimal(marketOutput);
currentCow = backupNextCow;
stopDay=true;
}
else{
userHerd->DeleteAnimal(marketOutput);
stopDay=false;
}
}
if (stopDay == false){
marketOutput = userMarket->SellStockerCalves(currentCow);
if (marketOutput != -1) {
    if (marketOutput == currentCow->ID) {
        userHerd->DeleteAnimal(marketOutput);
        currentCow = backupNextCow;
    } else {
        userHerd->DeleteAnimal(marketOutput);
    }
}

if (currentCow != backupNextCow) {
    currentCow->AGE++;
    if (currentCow->PREG == 1) {
        currentCow->DPREG++;
    } if (currentCow->LACT == 1) {
        currentCow->LDAY++;
    } currentCow = currentCow->next;
} else {
    currentCow=backupNextCow;
}
else {
    currentCow=currentCow->next;
}

DAY++;
currentDATE++;
simulationDAY++;
}

if (YEAR == 1) {
    out.open("report1.txt", ios::out, filebuf::sh_read);
    userMarket->BudgetReport(out);
    out.close();
    if (AfxMessageBox("Hit OK to Proceed to Year 1", IDOK) ==
        IDOK)
    _spawnlp(_P_WAIT, "notepad.exe", " report1.txt", NULL);
    if (userHerd->UseYearOption(2) == -1) {
        YEAR = 20;
    }
}
else if (YEAR == 2) {
    out.open("report2.txt", ios::out, filebuf::sh_read);
    userMarket->BudgetReport(out);
    out.close();
    if (AfxMessageBox("Hit OK to Proceed to Year 2", IDOK) ==
IDOK) { _spawnlp(_P_WAIT, "notepad.exe", " report2.txt", NULL);
    if (userHerd->UseYearOption(3) == -1){
        YEAR = 20;
    }
    }
}
else if (YEAR == 3){
    out.open("report3.txt", ios::out, filebuf::sh_read);
    userMarket->BudgetReport(out);
    out.close();
    if (AfxMessageBox("Hit OK to Proceed to Year 3", IDOK) == IDOK)
        _spawnlp(_P_WAIT, "notepad.exe", " report3.txt", NULL);
    AfxMessageBox("Hit OK to Finish Simulation", IDOK);
    }
userForage->ResetForageVariables();
userNutrit->ResetNutritionVariables();
userHerd->ResetCowVariables();
userMarket->ResetMarketVariables();
}
userHerd->DeleteHerd();
userHerd->ResetCowConCntr();
userForage->ResetForageConCntr();
if (cp->DestroyWindow())
    delete cp;
}
#endif
if (test.close();
#endif
delete mainmenu;
delete userHerd;
delete userForage;
delete userMarket;
delete userNutrit;
Cow.cpp
#include "stdafx.h"
#include "HerdSim.h"
#include "HerdSimDlg.h"
#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include "Cow.h"
#include "tools.h"
int Sterile;
int NumStkrYEnd;
int NumHeifers;
int NumCows;
int CyclHeifer1;
int CyclHeifer2;
int CyclHeifer3;
int CyclHeifer21;
int CyclHeifer22;
int CyclHeifer23;
int PregHef1;
int PregHef2;
int PregHef3;
int PregHef21;
int PregHef22;
int PregHef23;
int CyclCow1;
int CyclCow2;
int CyclCow3;
int CyclCow21;
int CyclCow22;
int CyclCow23;
int PregCow1;
int PregCow2;
int PregCow3;
int PregCow21;
int PregCow22;
int PregCow23;
int Dystocia2;
int Dystocia3;
int Dystocia4;
int CalfCows;
int CalfCows1;
int ARTI;
int TFWEANAGE;
int TSWEANAGE;
double AVFWEANAGE;
double AVSWEANAGE;
int FCALF;
int SCALF;
int DRET;
int CONCE;
int CONCE1;
extern int currentDATE;
extern int YEAR;
extern int simulationDAY;
int TSTOP;
char IONO;
int DNUMH;
int DNUMHS;
int DNUMS;
int DNUMSS;
int DNUMF;
int DNUM;
int DNUMR;
int CNUMB;
int CNUMBH;
int CNUMBS;
int CNUMS;
double TBWT;
double TBWTH;
double TBWTS;
double AVGBWT;
double AVGBWTH;
double AVGBWTS;
int CONUM;
int COWS;
int BreedingCows;
int CulledCows;
int BreedingReplacement;
int IMPH;
int IMPS;
int IMPSS;
int IMPHS;
int NBULLS;
char NIPSS;
char NIPSH;
int NumAvailRepla;
int PNUM;
int PTEST;
double CNRAT;
int ACDAT;
double CRATE;
double WRATE;
double CULRAT;
int SELEC;
int Servi;
int Servil;
double AvServConcp;
CString FARM;
CString ADDR;
CString FOWN;
CString FMAN;
CString LMAN;
extern int BUSE;
extern double BCST;
int TNCOH;
int TNSCOH;
int PHEF;
int POHEF;
extern double TWSS1;
extern double TWHS1;
extern double NHFC1;
extern double NSTC1;
int NumBreedSeasns;
char SpringCalfSeasn;
char FallCalfSeasn;
int FDBS;
int SEAS;
int LDBS;
int LDHBS;
int HSEAS;
int FDBHS;
int FDBS2;
int SEAS2;
int LDBS2;
int LDHBS2;
int HSEAS2;
int FDBHS2;
int WEANSDATE;
int WEANFDATE;
int BCScore[10];
int AgeBreeders[15];
double AvGestLen;
double TotGestLen;
int CowsFailWean;
int CowsFail;
int CowsAbort;
int CowsElig;
int EligCowsOpen;
int CowsInElig;
int CowsElig1;
int CowsInElig1;
int NumCowsYend;
int TOTALAGE;
int calcuADG;
int calcuseasADG;
double antiADG;
double antiADG1;
double antiADG2;
double antiADG3;
double antiADG4;

Cow::Cow()
{
}
ACDAT=0;
ADDR="Montgomery County, Virginia";
calcuADG=1;
antiADG=1.2;
calcuseasADG=0;
antiADG1=1.0;
antiADG2=1.6;
antiADG3=1.2;
antiADG4=1.0;
ARTI=0;
AvServConcp=0;
AWNWT=0;
BSAL=0.60;
BuyBreed=0;
BUYP=1;
BuyPure=1;
BUYRC=1;
CalfCows=0;
CalfCowsl=0;
CLNUM=0;
COLR=1;
CONCE=0;
CONCE1=0;
CONUM=51;
COWS=CONUM-1;
BreedingCows=COWS;
BreedingReplacement=0;
CulledCows=0;
CNRAT=0;
CNUMB=0;
CNUMBH=0;
CNUMBS=0;
CULAG=15;
CULPO=1;
CyclHeifer1=0;
CyclHeifer2=0;
CyclHeifer3=0;
CyclHeifer21=0;
CyclHeifer22=0;
CyclHeifer23=0;
PregHef1=0;
PregHef2=0;
PregHef3=0;
PregHef21=0;
PregHef22=0;
PregHef23=0;
CyclCow1=0;
CyclCow2=0;
CyclCow3=0;
CyclCow21=0;
CyclCow22=0;
CyclCow23=0;
PregCow1=0;
PregCow2=0;
PregCow3=0;
PregCow21=0;
PregCow22=0;
PregCow23=0;
Sterile=0;
DNUM=0;
DNUMF=0;
DNUMH=0;
DNUMHS=0;
DNUMR=0;
DNUMS=0;
DNUMSS=0;
DRET=120;
Dystocia2=0;
Dystocia3=0;
Dystocia4=0;
FARM="BlueBerry Ridge Farms";
FallCalfSeasn=0;
FDBS=167;
FDBS2=335;
FDBHS=142;
FDBHS2=310;
fdbsday=16;
fdbsday2=1;
fdbsmonth=6;
fdbsmonth2=12;
earlyday=22;
earlyday2=6;
earlymonth=5;
earlymonth2=11;
FMAN="Ms. Shelly Doe";
FOWN="Mr. W. D. Doe";
EZCALF=1;
EZBREED=1;
HBBREED=7;
HBREED=1;
HSEAS=0;
HSEAS2=0;
IMPH=0;
IMPS=1;
IMPHS=1;
IMPSS=1;
IONO=0;
LMAN="Ms Catherine Doe";
NBULLS=2;
NCTB=0;
NHFC=0;
NIPSH=1;
NIPSS=2;
NumBreedSeasns=1;
NSHFC=0;
NSTC=0;
NSSTC=0;
NumAvailRepla=0;
NumHeifers=0;
NumCows=0;
PHEF=0;
POHEF=0;
PNUM=0;
PTEST=60;
PURE=1;
REPLF=0;
REPLR=0;
SEAS=90;
SEAS2=90;
LDBS=FDBS+SEAS;
LDBS2=FDBS2+SEAS2;
LDHBS=FDBHS+SEAS;
LDHBS2=FDBHS2+SEAS2;
SEASON=1;
SELEC=0;
SelectWILHerd=0;
Servi=0;
Servil=0;
SpringCalfSeasn=0;
TCDAT=0;
TEST=0;
TNCW=0;
TSTOP=0;
TWHS=0;
TWSS=0;
TBWT=0;
TBWTS=0;
AVGBWTS=0;
TBWTH=0;
AVGBWTH=0;
AvGestLen=0;
TotGestLen=0;
CowsFail=0;
CowsFailWean=0;
CowsAbort=0;
CowsElig1=0;
CowsInElig1=0;
weansday=5;
weansmonth=11;
weanfday=2;
weanfmonth=6;
WEANSDATE=309;
WEANFDATE=153;
WNWT=0;
TFWEANAGE=0;
TSWEANAGE=0;
AVFWEEANAGE=0;
AVSWEANAGE=0;
FCALF=0;
SCALF=0;
PER[0]=0;
PER[1]=.5;
PER[2]=0;
PER[3]=.5;
PER[4]=0;
PER[5]=0;
PER[6]=0;
PER[7]=0;
PER[8]=0;
PER[9]=0;
PER[10]=0;
PER[11]=0;
PER[12]=0;
BULBREED[0]=0;
BULBREED[1]=1;
BULBREED[2]=2;
BULBREED[3]=3;
BULBREED[4]=4;
BULBREED[5]=5;
BULBREED[6]=6;
BULBREED[7]=7;
BULBREED[8]=8;
BULBREED[9]=9;
BULBREED[10]=10;
BULBREED[12]=12;
BCScore[0] = 0;
BCScore[1] = 0;
BCScore[2] = 0;
BCScore[3] = 0;
BCScore[4] = 0;
BCScore[5] = 0;
BCScore[6] = 0;
BCScore[7] = 0;
BCScore[8] = 0;
BCScore[9] = 0;
AgeBreeders[0]=0;
AgeBreeders[1]=0;
AgeBreeders[2]=0;
AgeBreeders[3]=0;
AgeBreeders[4]=0;
AgeBreeders[5]=0;
AgeBreeders[6]=0;
AgeBreeders[7]=0;
AgeBreeders[8]=0;
AgeBreeders[9]=0;
AgeBreeders[10]=0;
AgeBreeders[11]=0;
AgeBreeders[12]=0;
AgeBreeders[13]=0;
AgeBreeders[14]=0;
NumCowsYend=0;
NumStkrYEnd=0;
TOTALAGE=0;
}

int Cow::UseYearOption(int year)
{
    YearOption* input = new YearOption;
    CString* newCaption = new CString;
    newCaption->Format("Simulation - Change for Year %i", year);
    input->m_caption = *newCaption;
    int response = input->DoModal();
    if (response == IDCANCEL){
        delete input;
        return -1;
    }
    if (input->m_changesire == true){
        HBBREED = input->m_sire + 1 ;
    }
    if (input->m_changeezsire == true){
        EZBREED = input->m_sire2 + 1 ;
    }
    delete input;
    return 0;
}

void Cow::UseInput1()
{
    CInput1* input = new CInput1;
    input->m_address=ADDR;
    input->m_farmman=FMAN;
    input->m_farmowner=FOWN;
    input->m_liveman=LMAN;
    input->m_farmname=FARM;
    int response=input->DoModal();
    if (response == IDCANCEL){
        delete input;
        return;
    }
    FARM=input->m_farmname;
    ADDR=input->m_address;
    FOWN=input->m_farmowner;
    FMAN=input->m_farmman;
    LMAN=input->m_liveman;
    delete input;
if (response == IDC_NEXT){
    UseInput2();
} return;

void Cow::UseInput2(){
    CInput2* input = new CInput2;
    input->m_AI=ARTI;
    input->m_buyreg=BUYP;
    input->m_buycalves=BUYRC;
    input->m_conum=CONUM-1;
    input->m_costbull=BCST;
    input->m_black=COLR;
    input->m_cullage=CULAG-8;
    input->m_cullpol=CULPO;
    input->m_implantheifer=IMPH;
    input->m_implantsheifer=IMPHS;
    input->m_implantssteer=IMPSS;
    input->m_implantsteer=IMPS;
    input->m_iono=IONO;
    input->m_numbulls=NBULLS;
    input->m_salvbull=BSAL;
    input->m_selecwiherd=SelectWIHerd;
    input->m_shimplants=NIPSH;
    input->m_yearsbull=BUSE;
    input->m_repbreed=BuyBreed;
    input->m_reppure=BuyPure;
    int response=input->DoModal();
    if (response == IDCANCEL){
        delete input;
        return;
    }
    ARTI=input->m_AI;
    BUYP=input->m_buyreg;
    BUYRC=input->m_buycalves;
    CONUM=input->m_conum+1;
    COWS=CONUM-1;
    BCST=input->m_costbull;
    CULAG=8 + input->m_cullage;
    CULPO=input->m_cullpol;
    COLR=input->m_black;
    IMPH=input->m_implantheifer;
    IMPHS=input->m_implantsheifer;
    IMPSS=input->m_implantssteer;
    IMPS=input->m_implantsteer;
    IONO=input->m_iono;
    NBULLS=input->m_numbulls;
    BSAL=input->m_salvbull;
    NIPSH=input->m_shimplants;
}
NIPSS=input->m_ssimplants;
SelectWIHerd=input->m_selecwiherd;
if ((SelectWIHerd == 0) && (BUYRC==0)){
    BUYRC=1;
}
BUSE=input->m_yearsbull;
BuyBreed=input->m_repbreed;
BuyPure=input->m_reppure;
BreedingCows=COWS;
delete input;
if (response == IDC_NEXT){
    UseInput11();
}
else if (response == IDC_PREV){
    UseInput1();
}
return;

void Cow::UseInput11()
{
    Cinput11* input = new Cinput11;
input->m_retstocker=DRET;
ininput->m_adg1=calcuADG;
ininput->m_antiadg=antiADG;
ininput->m_seasonal=calcuseasADG;
ininput->m_antiadg1=antiADG1;
ininput->m_antiadg2=antiADG2;
ininput->m_antiadg3=antiADG3;
ininput->m_antiadg4=antiADG4;
    int response=input->DoModal();
    if (response == IDCANCEL){
        delete input;
        return;
    }
    DRET=input->m_retstocker;
    calcuADG=input->m_adg1;
    antiADG=input->m_antiadg;
    calcuseasADG=input->m_seasonal;
    antiADG1 = input->m_antiadg1;
    antiADG2 = input->m_antiadg2;
    antiADG3 = input->m_antiadg3;
    antiADG4 = input->m_antiadg4;
delete input;
    if (response == IDC_NEXT){
        UseInput2A();
    }
    else if (response == IDC_PREV){
        UseInput2();
    }
return;
void Cow::UseInput2A(){
CInput2A* input = new CInput2A;
input->m_angus=PER[1];
input->m_bred=PURE;
input->m_ezcalf=EZCALF;
input->m_ezbullbreed=EZBREED - 1;
input->m_bullbreed=HBBREED-1;
input->m_hereford=PER[2];
input->m_phereford=PER[3];
input->m_shorthorn=PER[4];
input->m_charol=PER[5];
input->m_chianina=PER[6];
input->m_gelb=PER[7];
input->m_limousin=PER[8];
input->m_main=PER[9];
input->m_pinz=PER[10];
input->m_pure=HBREED-1;
input->m_simmental=PER[11];
input->m_tarentaise=PER[12];
int response = input->DoModal();
if (response == IDCANCEL){
delete input;
return;
}
TRACE("m_bred: %i\n", input->m_bred);
PER[1]=input->m_angus;
PURE=input->m_bred;
EZCALF=input->m_ezcalf;
EZBREED=input->m_ezbullbreed + 1;
HBBREED=input->m_bullbreed + 1;
PER[2]=input->m_hereford;
PER[3]=input->m_phereford;
PER[4]=input->m_shorthorn;
PER[5]=input->m_charol;
PER[6]=input->m_chianina;
PER[7]=input->m_gelb;
PER[8]=input->m_limousin;
PER[9]=input->m_main;
PER[10]=input->m_pinz;
HBREED=input->m_pure + 1;
PER[11]=input->m_simmental;
PER[12]=input->m_tarentaise;
delete input;
TRACE("BRED: %i\n", PURE);
if (response == IDC_PREV){
UseInput11();
}
if (response == IDC_NEXT){
UseInput2B();
}
void Cow::UseInput2B(){
CInput2B* input = new CInput2B;
if (((SpringCalfSeasn==1) && (FallCalfSeasn==0))
input->m_breedingseasons = 0;
else if (((SpringCalfSeasn==0) && (FallCalfSeasn==1))
input->m_breedingseasons = 1;
else if (((SpringCalfSeasn==1) && (FallCalfSeasn==1))
input->m_breedingseasons = 2;
}
input->m_springmonth = fdbsmonth;
input->m_springday = fdbsday;
input->m_springlength = SEAS;
input->m_fallmonth = fdbsmonth2;
input->m_fallday = fdbsday2;
input->m_falllength = SEAS2;
input->m_shbse = HSEAS;
input->m_shbse2 = HSEAS2;
input->m_shbsemonth = earlymonth;
input->m_shbseday = earlyday;
input->m_shbse2month = earlymonth2;
input->m_shbse2day = earlyday2;
input->m_weansday=weansday;
input->m_weansmonth=weansmonth;
input->m_weanfday=weanfday;
input->m_weanfmonth=weanfmonth;
int response=input->DoModal();
if (response == IDCANCEL){
delete input;
return;
}
SEASON = input->m_breedingseasons+1;
if (SEASON==1){
SpringCalfSeasn=1;
FallCalfSeasn=0;
}
else if (SEASON==2){
FallCalfSeasn=1;
SpringCalfSeasn=0;
}
else if (SEASON==3){
FallCalfSeasn=1;
SpringCalfSeasn=1;
}
fdbsmonth = input->m_springmonth;
fdbsday = input->m_springday;
FDBS = Julian(fdbsmonth, fdbsday);
SEAS = input->m_springlength;
LDBS = FDBS + SEAS;
fdbsmonth2 = input->m_fallmonth;
fdbsday2 = input->m_fallday;
FDBS2 = Julian(fdbsmonth2, fdbsday2);
SEAS2 = input->m_falllength;
LDBS2 = FDBS2 + SEAS2;
HSEAS = input->m_shbse;
HSEAS2 = input->m_shbse2;
earlymonth = input->m_shbsemonth;
earlyday = input->m_shbseday;
earlymonth2 = input->m_shbse2month;
earlyday2 = input->m_shbse2day;
FDBHS = Julian(earlymonth, earlyday);
FDBHS2 = Julian(earlymonth2, earlyday2);
weansday = input->m_weansday;
weansmonth = input->m_weansmonth;
weanfday = input->m_weanfday;
weanfmonth = input->m_weanfmonth;
WEANSDATE = Julian(weansmonth, weansday);
WEANFDATE = Julian(weanfmonth, weanfday);
delete input;
if (response == IDC_PREV){
    UseInput2A();
}
return;
}

void Cow::Init()
{
    UseInput1();
}

void Cow::Abort(AnimalRecord *animal){
double ABORT=0;
double FBORT=0;
int OLD=0;
double randomnumber = (double) rand();
double MCBORT = (randomnumber/(double) RAND_MAX);
if ( (animal->SEX == 3) && (animal->PREG == 1) ){
    OLD = animal->AGE / 365;
    ABORT = 0.115 + (0.01 * OLD);
    FBORT = (ABORT * 0.03) / (animal->PGEST - 40);
    if (animal->DPREG <= 16){
        if (MCBORT <= ((ABORT *.6111) / 16)){
            animal->PREG=0;
            animal->DPREG=0;
            animal->PGEST=0;
            DeleteAnimal(animal->FIDN);
            animal->FIDN=0;
        }
    }
}
```c
animal->CSEX=0;
animal->CBWT=0;
PNUM--;  
CONCE--; 
}
}
else if ((animal->DPREG > 16) && (animal->DPREG <= 40)) {
  if (MCBORT <= ((ABORT * .3589) / 24)) {
    animal->CUCYC+=animal->DPREG;
animal->PREG=0;
animal->DPREG=0;
animal->PGEST=0;
DeleteAnimal(animal->FIDN);
animal->FIDN=0;
animal->CSEX=0;
animal->CBWT=0;
PNUM--; 
  }
}
else if ((animal->DPREG > 40) && ((animal->DPREG) < (animal->PGEST ))) {
  if( (MCBORT <= FBORT) && ((animal->CUCYC + animal->DPREG)<=365) ){
    animal->CUCYC+=animal->DPREG;
animal->PREG=0;
animal->DPREG=0;
animal->PGEST=0;
DeleteAnimal(animal->FIDN);
animal->FIDN=0;
animal->CSEX=0;
animal->CBWT=0;
PNUM--; 
  if ( (currentDATE>(LDBS + PTEST)) && (SpringCalfSeasn == 1) 
  && (currentDATE <= 365) ){ 
    CowsAbort++;
  }
  else if ( (currentDATE>(LDBS + PTEST)) && (SpringCalfSeasn == 1) && (currentDATE >= 365) && (currentDATE <= FDBS) ){
    CowsAbort++;
  }
  else if ( ((LDBS2 + PTEST)<=365) && (FallCalfSeasn == 1) && (currentDATE > (LDBS2 + PTEST)) && (currentDATE<=365) ){
    CowsAbort++;
  }
  else if ( ((LDBS2 + PTEST)>365) && (FallCalfSeasn == 1) && (currentDATE>=(LDBS2 + PTEST) - 365) && (currentDATE<=FDBS2) ){
    CowsAbort++;
  }
  }
else if ( (MCBORT <= FBORT) && ( ((animal->CUCYC) + (animal->DPREG) <= (365) ) ){
    CowsAbort++;
  }
```
if ((LDBS2 + PTEST) <= 365) && (FallCalfSeasn == 1) && 
   (currentDATE > (LDBS2 + PTEST)) && (currentDATE <= 365) 
   && (animal->PREG == 0) 
   )
   
animal->FAIL++;
CowsFail++;
}

if (animal->PREG == 0)
   
animal->CDWTK = ((animal->SBWTK - (animal->GUWTK))/0.96);
   animal->GUWTK = 0;
   animal->CDWT = animal->CDWTK * 2.20459;
}

if (NumBreedSeasns==1)
   
   if (SpringCalfSeasn==1)
   
   if ( (currentDATE > (LDBS + PTEST)) && (currentDATE <= 365) 
      && (animal->PREG == 0) 
   )
   
animal->FAIL++;
CowsFail++;
}

else if ( (currentDATE==1) && (currentDATE<=FDBS) && (animal->PREG==0) )
   
animal->FAIL++;
CowsFail++;
}

else if (FallCalfSeasn==1)
   
   if ( ((LDBS2 + PTEST)<=365) && (currentDATE > (LDBS2 + PTEST)) 
      && (currentDATE<=365) 
   )
   
   CowsAbort++;
}

else if ( (currentDATE>(LDBS + PTEST)) && (SpringCalfSeasn == 1) 
   && (currentDATE <= 365) 
   )
   
   CowsAbort++;
}

else if ( (currentDATE>(LDBS + PTEST)) && (SpringCalfSeasn == 1) 
   && (currentDATE >= 365) && (currentDATE <=FDBS) )
   
   CowsAbort++;
}

if (NumBreedSeasns==1)
   
   if (SpringCalfSeasn==1)
   
   if ( (currentDATE > (LDBS + PTEST)) 
      && (currentDATE <= 365) 
   )
   
   CowsAbort++;
}

if (currentDATE>(LDBS + PTEST)) 
   
   animal->CUCYC= currentDATE;
   animal->PREG=0;
   animal->DPREG=0;
   animal->PGEST=0;
   DeleteAnimal(animal->FIDN);
   animal->FIDN=0;
   animal->CSEX=0;
   animal->CBWT=0;
   PNUM--;
CowsFail++;
}
else if ( ((LDBS2 + PTEST)>365) && (currentDATE>=(LDBS2 + PTEST) - 365)) && (currentDATE<=FDBS2) && (animal->PREG==0) ){
    animal->FAIL++;
    CowsFail++;
}
}
else if (NumBreedSeasns==2){
    if ( (currentDATE >= (LDBS + PTEST)) && (currentDATE<=FDBS2) && (animal->PREG == 0) ){
        animal->SEAOP1=1;
    }
    else if ( ((LDBS2 + PTEST)<=365) && (currentDATE > (LDBS2 + PTEST)) && (currentDATE<=365) && (animal->PREG==0) ){
        animal->SEAOP1=1;
        if ( ((animal->SEAOP1 + animal->SEAOP2) == 2) ){
            animal->FAIL++;
            CowsFail++;
        }
    }
    else if ( ((LDBS2 + PTEST)>365) && (currentDATE>((LDBS2 + PTEST) - 365)) && (currentDATE<=FDBS) && (animal->PREG==0)) {
        animal->SEAOP2=2;
        if ( ((animal->SEAOP1 + animal->SEAOP2) == 2) ){
            animal->FAIL++;
            CowsFail++;
        }
    }
}
}

void Cow::BirthAnimal(AnimalRecord *animal, AnimalRecord *mother){
    int j;
    double BBWT[13]={0,
                     80.9,86.2,86.2,85.8,90.6,89.3,86.0,85.8,90.6,86.4,88.9,82.7} ;
    double BPADG[13]={0, 1.87, 1.70, 1.70, 1.80, 1.85, 1.83, 1.87, 1.76, 1.81, 1.76, 1.83, 1.81} ;
    double BPMWT[13]={0, 1156, 1188, 1188, 1188, 1391, 1563, 1294, 1211, 1521, 1232, 1296, 1134} ;
    double BPMSWT[13]={0, 969, 1038, 1038, 988, 1238, 1350, 1175, 1113, 1238, 1075, 1213, 1062} ;
    double BPGAVG[13]={0,285.4, 188.6, 288.6, 288.6, 285.4, 286.3, 288.1, 285.4, 286.3, 288.6, 287.5, 286.3} ;
    double randomnumber=rand();
}
double MCSEX = (randomnumber/(double) RAND_MAX);
double BWTAVG;
double BSD;
double PADGAVG;
double PADGSD;
double PMTAVG;
double PMSD;
double PMSSD;
double PMSTAVG;
double SEXX = 0.5;
if (MCSEX <= SEX)
    animal->SEX = 2;
mother->CSEX = 2;
else
    animal->SEX = 0;
mother->CSEX = 0;
if (PURE==0)
    animal->SIREBREED=mother->BREED;
else if ((PURE==1) && (mother->PARIE>=2))
    animal->SIREBREED=HBBREED;
else if ((PURE==1) && (mother->PARIE<2) && (EZCALF==0))
    animal->SIREBREED=HBBREED;
else if ((PURE==1) && (mother->PARIE<2) && (EZCALF==1))
    animal->SIREBREED=EZBREED;
animal->ID=LastID()+1;
mother->FIDN=animal-> ID;
animal->MIDN=mother-> ID;
animal->BCS=5;
animal->BREED=0;
animal->BWT=0;
animal->BIRTHDATE=0;
animal->ACBWT=0;
animal->CADAT=0;
animal->CALFSEAS=0;
animal->CBWT=0;
animal->CDWT=0;
animal->CDWTK=0;
animal->CIDN=0;
animal->CBIRTHDATE=0;
animal->CNDAT=0;
animal->COLR=COLR;
animal->CSEX=0;
animal->CUCYC=0;
animal->CULL=0;
animal->DETEC=0;
animal->DIE=0;
animal->DPREG=0;
animal->DYSTO=0;
animal->EBWTK=0;
animal->EQSBW=0;
animal->EQEBW=0;
animal->FAIL=0;
animal->FIDN=0;
animal->FWEAN=0;
animal->GUWTK=0;
animal->IMPLANT=0;
animal->LACT=0;
animal->LDAY=0;
animal->MCMFC=0;
animal->MCSNF=0;
animal->PARIE=0;
animal->PAST=1;
animal->PGEST=0;
animal->PPI=0;
animal->PREG=0;
animal->PUBER=0;
animal->INELIG=1;
animal->Puberty=0;
animal->PURE=PURE;
animal->REPLA=0;
animal->SBWTK=0;
animal->WEAN=0;
animal->WEANDATE=0;
animal->WEANWT=0;
animal->DHETBWT=0;
animal->DHETADG=0;
animal->DHETMWT=0;
animal->MHETADG=0;
animal->MHETBWT=0;
animal->SEAOP=0;
animal->SEAOP1=0;
animal->SEAOP2=0;
animal->SellCalf=0;
animal->SERVI=0;
animal->UNPREG=0;
animal->YN=0;
animal->YEN=0;
animal->AGE=0;
animal->SOUR = 0;
animal->AGED=0;
animal->PGAVG=0;
animal->PGSD=0;
animal->Fgeneration=0;
animal->GGGSIREBREED=0;
animal->GGSIREBREED=0;
animal->GSIREBREED=0;
if ((YEAR <= 3) && (PURE == 0) && ((animal->DAMBREED) == (animal->SIREBREED)) && (animal->SEX == 0)) {
if (animal->SIREBREED == 1) {
animal->CDWT = GetStochasticNumber(9.91, 80.9);
animal->PMWT = GetStochasticNumber(139.96, 1156);
animal->BWT = animal->CDWT;
animal->ACBWT = 80.9;
animal->PADG = GetStochasticNumber(0.160, 1.87);
animal->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT = GetStochasticNumber(26.0, 969);
animal->PGA VG = 285.4;
animal->PGSD = 6.02;
animal->BREED = animal->SIREBREED;
} else if (animal->SIREBREED == 2) {
animal->CDWT = GetStochasticNumber(10.56, 86.2);
animal->PMWT = GetStochasticNumber(143.84, 1188);
animal->BWT = animal->CDWT;
animal->ACBWT = 86.2;
animal->PADG = GetStochasticNumber(0.145, 1.70);
animal->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT = GetStochasticNumber(27.85, 1038);
animal->PGA VG = 288.6;
animal->PGSD = 6.09;
animal->BREED = animal->SIREBREED;
} else if (animal->SIREBREED == 3) {
animal->CDWT = GetStochasticNumber(10.56, 86.2);
animal->PMWT = GetStochasticNumber(143.84, 1188);
animal->BWT = animal->CDWT;
animal->ACBWT = 86.2;
animal->PADG=GetStochasticNumber(0.145, 1.70);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT=GetStochasticNumber(27.85, 1038);
animal->PGAVG=288.6;
animal->PGSD=6.09;
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 4){
animal->CDWT=GetStochasticNumber(10.51, 85.8);
animal->PMWT=GetStochasticNumber(143.84, 1188);
animal->BWT =animal->CDWT;
animal->ACBWT=85.8;
animal->PADG=GetStochasticNumber(0.154, 1.8);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT=GetStochasticNumber(26.51, 988);
animal->PGAVG=288.6;
animal->PGSD=6.09;
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 5){
animal->CDWT=GetStochasticNumber(11.1, 90.6);
animal->PMWT=GetStochasticNumber(168.42, 1391);
animal->BWT =animal->CDWT;
animal->ACBWT=90.6;
animal->PADG=GetStochasticNumber(0.158, 1.85);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT=GetStochasticNumber(33.218, 1238);
animal->PGAVG=285.9;
animal->PGSD=6.03;
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 6){
animal->CDWT=GetStochasticNumber(10.94 , 89.3);
animal->PMWT=GetStochasticNumber(189.24, 1563);
animal->BWT =animal->CDWT;
animal->ACBWT=89.3;
animal->PADG=GetStochasticNumber(0.157 , 1.83);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT=GetStochasticNumber(36.223, 1350);
animal->PGAVG=287.5;
animal->PGSD=6.07;
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 7){
animal->CDWT=GetStochasticNumber(10.54 , 86.0);
animal->PMWT=GetStochasticNumber(156.67, 1294);
animal->BWT =animal->CDWT;
animal->ACBWT=86.0;
animal->PADG=GetStochasticNumber(0.160 , 1.87);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT=GetStochasticNumber(31.527, 1175);
animal->PGAVG=286.3;
animal->PGSD=6.04;
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 8){
animal->CDWT=GetStochasticNumber(10.51, 85.8);
animal->PMWT=GetStochasticNumber(146.62, 1211);
animal->BWT =animal->CDWT;
animal->ACBWT=85.8;
animal->PADG=GetStochasticNumber(0.151 , 1.76);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT=GetStochasticNumber(29.864, 1113);
animal->PGAVG=288.1;
animal->PGSD=6.08;
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 9){
animal->CDWT=GetStochasticNumber(11.1, 90.6);
animal->PMWT=GetStochasticNumber(184.16, 1521);
animal->BWT =animal->CDWT;
animal->ACBWT=90.6;
animal->PADG=GetStochasticNumber(0.154, 1.81);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT=GetStochasticNumber(33.218, 1238);
animal->PGAVG=285.4;
animal->PGSD=6.02;
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 10){
animal->CDWT=GetStochasticNumber(10.58, 86.4);
animal->PMWT=GetStochasticNumber(149.17, 1232);
animal->BWT =animal->CDWT;
animal->ACBWT=86.4;
animal->PADG=GetStochasticNumber(0.151, 1.76);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMSWT=GetStochasticNumber(28.844, 1075);
animal->PGAVG=286.0;
animal->PGSD=6.03;
animal->BREED=animal->SIREBREED;
} else if (animal->SIREBREED == 11){
    animal->CDWT=GetStochasticNumber(10.89, 88.9);
    animal->PMWT=GetStochasticNumber(156.91, 1296);
    animal->BWT =animal->CDWT;
    animal->ACBWT=88.9;
    animal->PADG=GetStochasticNumber(0.157, 1.83);
    animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
    animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
    animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
    animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
    animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
    animal->PMSWT=GetStochasticNumber(32.547, 1213);
    animal->PGAVG=286.3;
    animal->PGSD=6.04;
    animal->BREED=animal->SIREBREED;
} else if (animal->SIREBREED == 12){
    animal->CDWT=GetStochasticNumber(10.13, 82.7);
    animal->PMWT=GetStochasticNumber(137.3, 1134);
    animal->BWT =animal->CDWT;
    animal->ACBWT=82.7;
    animal->PADG=GetStochasticNumber(0.154, 1.81);
    animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
    animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
    animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
    animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
    animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
    animal->PMSWT=GetStochasticNumber(28.495, 1062);
    animal->PGAVG=287.1;
    animal->PGSD=6.06;
    animal->BREED=animal->SIREBREED;
} }
else if ((YEAR <= 3) && (PURE == 0) && ((animal->DAMBREED)==(animal->SIREBREED))
    && (animal->SEX == 2))
    if( animal->SIREBREED == 1){
    animal->CDWT=GetStochasticNumber(9.91, 80.9);
    animal->PMWT=GetStochasticNumber(26.000, 969);
    animal->PMSWT =animal->PMWT;
    animal->BWT =animal->CDWT;
animal->ACBWT=80.9;
animal->PADG=GetStochasticNumber(0.160, 1.87);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 2){
animal->CDWT=GetStochasticNumber(10.56, 86.2);
animal->PMWT=GetStochasticNumber(27.850, 1038);
animal->PMSWT=animal->PMWT;
animal->BWT=animal->CDWT;
animal->ACBWT=86.2;
animal->PADG=GetStochasticNumber(0.145, 1.70);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 3){
animal->CDWT=GetStochasticNumber(10.56, 86.2);
animal->PMWT=GetStochasticNumber(27.850, 1038);
animal->PMSWT=animal->PMWT;
animal->BWT=animal->CDWT;
animal->ACBWT=86.2;
animal->PADG=GetStochasticNumber(0.145, 1.70);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 4){
animal->CDWT=GetStochasticNumber(10.51, 85.8);
animal->PMWT=GetStochasticNumber(26.510, 988);
animal->PMSWT =animal->PMWT;
animal->BWT =animal->CDWT;
animal->ACBWT=85.8;
animal->PADG=GetStochasticNumber(0.154 , 1.8 );
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 5){
animal->CDWT=GetStochasticNumber(11.1 , 90.6);
animal->PMWT=GetStochasticNumber(33.218, 1238);
animal->PMSWT =animal->PMWT;
animal->BWT =animal->CDWT;
animal->ACBWT=90.6;
animal->PADG=GetStochasticNumber(0.158 , 1.85);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
}
else if (animal->SIREBREED == 6){
animal->CDWT=GetStochasticNumber(10.94 , 89.3);
animal->PMWT=GetStochasticNumber(36.223, 1350);
animal->PMSWT =animal->PMWT;
animal->BWT =animal->CDWT;
animal->ACBWT=89.3;
animal->PADG=GetStochasticNumber(0.157 , 1.83);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
} else if (animal->SIREBREED == 7){
    animal->CDWT=GetStochasticNumber(10.54 , 86.0);
    animal->PMWT=GetStochasticNumber(31.527, 1175);
    animal->PMSWT =animal->PMWT;
    animal->BWT =animal->CDWT;
    animal->ACBWT=86.0;
    animal->PADG=GetStochasticNumber(0.160   , 1.87);
    animal->UADG1=GetStochasticNumber((antiADG/11.685), antiADG);
    animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
    animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
    animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
    animal->BREED=animal->SIREBREED;
} else if (animal->SIREBREED == 8){
    animal->CDWT=GetStochasticNumber(10.51 , 85.8);
    animal->PMWT=GetStochasticNumber(29.864, 1113);
    animal->PMSWT =animal->PMWT;
    animal->BWT =animal->CDWT;
    animal->ACBWT=85.8;
    animal->PADG=GetStochasticNumber(0.151  , 1.76);
    animal->UADG1=GetStochasticNumber((antiADG/11.685), antiADG);
    animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
    animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
    animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
    animal->BREED=animal->SIREBREED;
} else if (animal->SIREBREED == 9){
    animal->CDWT=GetStochasticNumber(11.1 , 90.6);
    animal->PMWT=GetStochasticNumber(33.218, 1238);
    animal->PMSWT =animal->PMWT;
    animal->BWT =animal->CDWT;
    animal->ACBWT=90.6;
    animal->PADG=GetStochasticNumber(0.154 , 1.81);
    animal->UADG1=GetStochasticNumber((antiADG/11.685), antiADG);
    animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
    animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
} else if (animal->SIREBREED == 10) {
animal->CDWT=GetStochasticNumber(10.58 , 86.4);
animal->PMWT=GetStochasticNumber(28.844 , 1075);
animal->PMSWT =animal->PMWT;
animal->BWT =animal->CDWT;
animal->ACBWT=86.4;
animal->PADG=GetStochasticNumber(0.151 , 1.76);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
} else if (animal->SIREBREED == 11) {
animal->CDWT=GetStochasticNumber(10.89 , 88.9);
animal->PMWT=GetStochasticNumber(32.547 , 1213);
animal->PMSWT =animal->PMWT;
animal->BWT =animal->CDWT;
animal->ACBWT=88.9;
animal->PADG=GetStochasticNumber(0.157 , 1.83);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->BREED=animal->SIREBREED;
} else if (animal->SIREBREED == 12) {
animal->CDWT=GetStochasticNumber(10.13 , 82.7);
animal->PMWT=GetStochasticNumber(28.495 , 1062);
animal->PMSWT =animal->PMWT;
animal->BWT =animal->CDWT;
animal->ACBWT=82.7;
animal->PADG=GetStochasticNumber(0.154 , 1.81);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), an-
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);

animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);

animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);

animal->BREED=animal->SIREBREED;

}

else if ((YEAR < 0) && (PURE != 0) && (animal->SEX == 0)) {

animal->BREED=mother->BREED;

animal->DAMBREED=mother->DAMBREED;

animal->SIREBREED=mother->SIREBREED;

animal->GSIREBREED=mother->GSIREBREED;

animal->GDAMBREED=mother->GDAMBREED;

BWTAVG =
((80.9*PER[1])+(86.2*PER[2])+(86.2*PER[3])+(85.8*PER[4])+(90.6*PER[5])
+(89.3*PER[6])+(86.0*PER[7])+(85.8*PER[8])+(90.6*PER[9])+(86.4*PER[10])
+(88.9*PER[11])+(82.7*PER[12]));

BSD= (.1225*BWTAVG);

animal->ACBWT=BWTAVG;

animal->BWT=GetStochasticNumber(BSD , BWTAVG);

animal->CDWT=animal->BWT;

PMTAVG =
((1156*PER[1])+(1188*PER[2])+ (1188*PER[3])+(1188*PER[4])+(1391*PER[5])
+(1563*PER[6])+(1294*PER[7])+(1211*PER[8])+(1521*PER[9])+(1232*PER[10])
+(1296*PER[11])+(1134*PER[12]));

PMSD = (.12105*PMTAVG);

animal->PMWT = GetStochasticNumber(PMSD, PMTAVG);

PADGAVG =
((1.87*PER[1])+(1.70*PER[2])+(1.70*PER[3])+(1.80*PER[4])+(1.85*PER[5])
+(1.83*PER[6])+(1.87*PER[7])+(1.76*PER[8])+(1.81*PER[9])+(1.76*PER[10])
+(1.83*PER[11])+(1.81*PER[12]));

PADGSD = (.11312*PADGAVG);

animal->PADG=GetStochasticNumber(PADGSD, PADGAVG);

animal->PGAVG =
((285.4*PER[1])+(288.6*PER[2])+(288.6*PER[3])+(288.6*PER[4])
+(285.9*PER[5])
+(287.5*PER[6])+(286.3*PER[7])+(288.1*PER[8])+(285.4*PER[9])
+(286.0*PER[10])
+(286.3*PER[11])+(287.1*PER[12]));

animal->PGSD = (.02110* animal->PGAVG);

PMSTAVG =
((969*PER[1])+(1038*PER[2])+(1038*PER[3])+(988*PER[4])+(1238*PER[5])
+(1350*PER[6])+(1175*PER[7])+(1113*PER[8])+(1238*PER[9])+(10
75*PER[10])
+(1213*PER[11])+(1062*PER[12]));
PMSSD =(.02683*PMSTAVG);
animal->PMSWT=GetStochasticNumber(PMSSD , PMSTAVG);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), an-
tiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), an-
tiADG4);
animal->Fgeneration=+1;
else if ((YEAR >= 0) && (PURE != 0) && (animal->SEX == 0)) {
for (j=1; j<13; j++){
if ((mother->PARIE>=2) || ( (mother->PARIE<2) &
EZCALF==0))){
if (BULBREED[j]==HBBREED){
BWTAVG=(((mother->BWT) - ((mother->DHETBWT) + (mother-
>MHETBWT)))/2);
BSD=(.1225*BWTAVG);
PMTAVG=(((mother->PMWT) - (mother-
>DHETMWT))+(BPMWT[j]))/2);
PMSSD=(.12105*PMTAVG);
PADGAVG=(((mother->PADG) - ((mother->DHETADG) + (mother-
>MHETADG))+(BPADG[j]))/2);
PADGSD=(.11312*PADGAVG);
PMSTAVG=(((mother->PMSWT)+(BPMSWT[j]))/2);
PMSSD=(.02683*PMSTAVG);
animal->ACBWT=BWTAVG;
animal->PGAVG=(((mother->PGAVG)+(BPGAVG[j]))/2);
animal->PGSD=(.02110* animal->PGAVG);
animal->BWT=GetStochasticNumber(BSD , BWTAVG);
animal->CDWT=animal->BWT;
animal->PADG=GetStochasticNumber(PADGSD, PADGAVG);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), an-
tiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), an-
tiADG4);
animal->PMWT=GetStochasticNumber(PMSD , PMTAVG);
animal->PMSWT=GetStochasticNumber(PMSSD , PMSTAVG);
animal->Fgeneration=(mother->Fgeneration + 1);}
if (animal->Fgeneration>0){
animal->GGSIREBREED=mother->GGSIREBREED;
animal->GGSIREBREED=mother->GSIREBREED;
animal->GSIREBREED=mother->SIREBREED;
animal->SIREBREED=HBBREED;
animal->BREED=HBBREED;
animal->GGGDAMBREED=mother->GGDAMBREED;
animal->GGDAMBREED=mother->GDAMBREED;
animal->GDAMBREED=mother->DAMBREED;
animal->DAMBREED=mother->BREED;
}
}
else if ( (mother->PARIE<2) && (EZCALF==1)) {
  if (BULBREED[j]==EZBREED) {
    BWTAVG=((((mother->BWT) - ((mother->DHETBWT) + (mother-
>MHETBWT))) + (BBWT[j]))/2);
    BSD=(.1225*BWTAVG);
    PMTAVG=((((mother->PMWT) - (mother-
>DHETMWT))) + (BPMWT[j]))/2);
    PMSD=(.12105*PMTAVG);
    PADGAVG=((((mother->PADG) - ((mother->DHETADG) + (mother-
>MHETADG))) + (BPADG[j]))/2);
    PADGSD=(.11312*PADGAVG);
    PMSTAVG=((((mother->PMSWT)+(BPMSWT[j]))/2);
    PMSSD=(.02683*PMSTAVG);
    animal->ACBWT=BWTAVG;
    animal->PGAVG=((((mother->PGAVG)+(BPGAVG[j]))/2);
    animal->PGSD=(.02110*animal->PGAVG);
    animal->BWT=GetStochasticNumber(BSD , BWTAVG);
    animal->CDWT=animal->BWT;
    animal->PADG=GetStochasticNumber(PADGSD, PADGAVG);
    animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
    animal->UADG1=GetStochasticNumber((antiADG1/11.685), an-
    tiADG1);
    animal->UADG2=GetStochasticNumber((antiADG2/11.685), an-
    tiADG2);
    animal->UADG3=GetStochasticNumber((antiADG3/11.685), an-
    tiADG3);
    animal->UADG4=GetStochasticNumber((antiADG4/11.685), an-
    tiADG4);
    animal->PMWT=GetStochasticNumber(PMSD , PMTAVG);
    animal->PMSWT=GetStochasticNumber(PMSSD , PMSTAVG);
    animal->Fgeneration=(mother->Fgeneration + 1);
  }
  if (animal->Fgeneration>0) {
    animal->GGSIREBREED=mother->GGSIREBREED;
    animal->GGSIREBREED=mother->GSIREBREED;
    animal->GSIREBREED=mother->SIREBREED;
    animal->SIREBREED=EZBREED;
    animal->BREED=EZBREED;
    animal->GGGDAMBREED=mother->GGDAMBREED;
    animal->GGDAMBREED=mother->GDAMBREED;
animal->GDAMBREED=mother->DAMBREED;
animal->DAMBREED=mother->BREED;
}
}
}

else if ((YEAR < 0) && (PURE != 0) && (animal->SEX == 2)) {
    for (j=1; j<13; j++){
        if ((mother->PARIE>=2) || ( (mother->PARIE<2) &&
            (EZCALF==0))){
            if (BULBREED[j]==HBBREED){
                BWTAVG=(((mother->BWT) - ((mother->DHETBWT) + (mother->
                    >MHETBWT)))+(BBWT[j])/2);
                BSD=(.1225*BWTAVG);
                PMTAVG=(((mother->PMSWT)+(BPMSWT[j]))/2);
                PMSD=(.02683*PMTAVG);
                PADGAVG=(((mother->PADG) - ((mother->DHETADG) + (mother->
                    >MHETADG)))+(BPADG[j])/2);
                PADGSD=(.11312*PADGAVG);
                animal->ACBWT=BWTAVG;
                animal->PAGAVG=(((mother->PGA+BAVG)+BPAGAVG[j])/2);
                animal->PGSD=(.02110*animal->PAGAVG);
                animal->BWT=GetStochasticNumber(BSD, BWTAVG);
                animal->CDWT=animal->BWT;
                animal->PADG=GetStochasticNumber(PADGSD, PADGAVG);
                animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
                animal->UADG1=GetStochasticNumber((antiADG1/11.685), an-
                    tiADG1);
                animal->UADG2=GetStochasticNumber((antiADG2/11.685), an-
                    tiADG2);
                animal->UADG3=GetStochasticNumber((antiADG3/11.685), an-
                    tiADG3);
                animal->UADG4=GetStochasticNumber((antiADG4/11.685), an-
                    tiADG4);
                animal->PMWT=GetStochasticNumber(PMSD, PMTAVG);
                animal->PMSWT=animal->PMWT;
                animal->BREED=HBBREED;
                animal->Fgeneration=(mother->Fgeneration + 1);
            }
        }
    }
    if (animal->Fgeneration>0){
        animal->GGGSIREBREED=mother->GGSIREBREED;
        animal->GGSIREBREED=mother->GSIREBREED;
        animal->ISIREBREED=mother->SIREBREED;
        animal->SIREBREED=HBBREED;
        animal->BREED=HBBREED;
        animal->GGDAMBREED=mother->GGDAMBREED;
        animal->GDAMBREED=mother->DAMBREED;
        animal->DAMBREED=mother->BREED;
    }
}
else if ( (mother->PARIE<2) && (EZCALF==1)) {
    if (BULBREED[j]==EZBREED) {
        BWTAVG=(((mother->BWT) - ((mother->DHETBWT) + (mother->MHETBWT))) + (BBWT[j]))/2);
        BSD=(.1225*BWTAVG);
        PMTAVG=(((mother->PMSWT)+(BPMSWT[j]))/2);
        PMSD=(.02683*PMTAVG);
        PADGAVG=(((mother->PADG) - ((mother->DHETADG) + (mother->MHETADG))) + (BPADG[j]))/2);
        PADGSD=(.11312*PADGAVG);
        animal->ACBWT=BWTAVG;
        animal->PGAVG=(((mother->PGA)+ (BPGA[j]))/2);
        animal->PGSD=(.02110*animal->PGAVG);
        animal->BWT=GetStochasticNumber(BSD , BWTAVG);
        animal->CDWT=animal->BWT;
        animal->PADG=GetStochasticNumber(PADGSD, PADGAVG);
        animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
        animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
        animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
        animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
        animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
        animal->PMWT=GetStochasticNumber(PMSD , PMTAVG);
        animal->PMSWT=animal->PMWT;
        animal->BREED=HBBREED;
        animal->Fgeneration=(mother->Fgeneration + 1);
    }
    if (animal->Fgeneration>0) {
        animal->GGGSIREBREED=mother->GGSIREBREED;
        animal->GSSIREBREED=mother->GSIREBREED;
        animal->SIREBREED=mother->SIREBREED;
        animal->BREED=EZBREED;
        animal->GGDAMBREED=mother->GGDAMBREED;
        animal->GDAMBREED=mother->DAMBREED;
        animal->DAMBREED=mother->BREED;
    }
}
else if (((YEAR >= 0) && (PURE != 0) && (animal->SEX == 2)) { for (j=1; j<13; j++){ if ((mother->PARIE>=2) || ( (mother->PARIE<2) && (EZCALF==0)))){
            if (BULBREED[j]==HBBREED) {
                BWTAVG=(((mother->BWT) - ((mother->DHETBWT) + (mother->MHETBWT))) + (BBWT[j]))/2;
            }
        }
    }
}
BSD= (.1225*BWTAVG);
PMTAVG= (((mother->PMSWT)+(BPMSWT[j]))/2);
PMSD=(.02683*PMTAVG);
PADGAVG= (((((mother->PADG)-(mother->DHETADG)) + (mother->MHETADG))+(BPADG[j]))/2);
PADGSD=(.11312*PADGAVG);
animal->ACBWT=BWTAVG;
animal->PGAVG=(((mother->PGAVG)+(BPGAVG[j]))/2);
animal->PGSD=(.02110*animal->PGAVG);
animal->BWT=GetStochasticNumber(BSD , BWTAVG);
animal->CDWT=animal->BWT;
animal->PADG=GetStochasticNumber(PADGSD, PADGAVG);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMWT=GetStochasticNumber(PMSD , PMTAVG);
animal->PMSWT=animal->PMWT;
animal->Fgeneration=(mother->Fgeneration + 1);
} if (animal->Fgeneration>0){
animal->GGGSIREBREED=mother->GGSIREBREED;
animal->GGSIREBREED=mother->GSIREBREED;
animal->GSIREBREED=mother->SIREBREED;
animal->SIREBREED=HBBREED;
animal->BREED=HBBREED;
animal->GGGDAMBREED=mother->GGDAMBREED;
animal->GGDAMBREED=mother->GDAMBREED;
animal->GDAMBREED=mother->DAMBREED;
animal->DAMBREED=mother->BREED;
}
else if ( (mother->PARIE<2) && (EZCALF==1)){
if (BULBREED[j]==EZBREED){
BWTAVG= (((((mother->BWT)-(mother->DHETBWT)) + (mother->MHETBWT)) + (BBWT[j]))/2);
BSD=(.1225*BWTAVG);
PMTAVG= (((mother->PMSWT)+(BPMSWT[j]))/2);
PMSD=(.02683*PMTAVG);
PADGAVG= (((((mother->PADG)-(mother->DHETADG)) + (mother->MHETADG))+(BPADG[j]))/2);
PADGSD=(.11312*PADGAVG);
animal->ACBWT=BWTAVG;
animal->PGAVG=(((mother->PGAVG)+(BPGAVG[j]))/2);
animal->PGSD=(.02110*animal->PGAVG);
animal->BWT=GetStochasticNumber(BSD , BWTAVG);
animal->CDWT=animal->BWT;
animal->PADG=GetStochasticNumber(PADGSD, PADGAVG);
animal->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
animal->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
animal->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
animal->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
animal->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
animal->PMWT=GetStochasticNumber(PMSD, PMTAVG);
animal->PMSTW=animal->PMWT;
animal->Fgeneration=(mother->Fgeneration + 1);
if (animal->Fgeneration>0){
animal->GGGSIREBREED=mother->GGSIREBREED;
animal->GGSIREBREED=mother->GSIREBREED;
animal->SIREBREED=EZBREED;
animal->GIDDAMBREED=mother->GIDDAMBREED;
animal->GIDDAMBREED=mother->GIDDAMBREED;
animal->GIDDAMBREED=mother->GIDDAMBREED;
animal->GIDDAMBREED=mother->BREED;
}
if (animal->Fgeneration==1){
animal->ACBWT=(animal->ACBWT*1.033);
animal->MHETBWT=(animal->BWT*.033);
animal->MHETADG=(animal->PADG*.08534);
} else if (animal->Fgeneration==2){
if (animal->GSIREBREED!=animal->GDAMBREED){
animal->ACBWT=(animal->ACBWT*1.033);
animal->MHETBWT=(animal->BWT*.033);
animal->MHETADG=(animal->PADG*.08534);
} else if ((animal->GSIREBREED==animal->GDAMBREED) && (animal->GSIREBREED==animal->GGSIREBREED) && (animal->GSIREBREED!=animal->GGDAMBREED)){
animal->ACBWT=(animal->ACBWT*(1+(.033*.5)));
animal->MHETBWT=(animal->BWT*(.033*.5));
animal->MHETADG=(animal->PADG*(.08534*.5));
} else if ((animal->GSIREBREED==animal->GDAMBREED) && ((animal->GSIREBREED!=animal->GGSIREBREED) && (animal->GSIREBREED==animal->GGDAMBREED))){
animal->ACBWT=(animal->ACBWT*(1+.033*.5));
} else if ((animal->GSIREBREED==animal->GDAMBREED) && ((animal->GSIREBREED!=animal->GGSIREBREED) && (animal->GSIREBREED==animal->GGDAMBREED))){
animal->ACBWT=(animal->ACBWT*(1+.033*.5));
}
animal->MHETBWT=(animal->BWT*(.033*.5));
animal->MHETADG=(animal->PADG*(.08534*.5));
}
}
else if ((animal->Fgeneration>=3) && ((animal->GSIREBREED!=animal->GDAMBREED) || (animal->GSIREBREED!=animal->GGSIREBREED) || (animal->GSIREBREED!=animal->GGDAMBREED) || (animal->GSIREBREED==animal->GGGDAMBREED))){
    animal->ACBWT=(animal->ACBWT*(1+(.033*.75)));
    animal->MHETBWT=(animal->BWT*(.033 *.75));
    animal->MHETADG=(animal->PADG*(.08534 * .75));
}
else if ((animal->GSIREBREED!=animal->GDAMBREED) && (animal->GSIREBREED!=animal->GGSIREBREED) && (animal->GSIREBREED==animal->GGDAMBREED))){
    animal->ACBWT=(animal->ACBWT*(1+(.033*.75)));
    animal->MHETBWT=(animal->BWT*(.033 * .75));
    animal->MHETADG=(animal->PADG*(.08534 * .75));
}
else if ((animal->GSIREBREED!=animal->GDAMBREED) && (animal->GSIREBREED==animal->GGSIREBREED) && (animal->GSIREBREED!=animal->GGDAMBREED)){
    animal->ACBWT=(animal->ACBWT*(1+(.033*.5)));
    animal->MHETBWT=(animal->BWT*(.033*.5));
    animal->MHETADG=(animal->PADG*(.08534*.5));
}
else if ((animal->GSIREBREED==animal->GDAMBREED) && ((animal->GSIREBREED!=animal->GGSIREBREED) || (animal->GSIREBREED!=animal->GGDAMBREED))){
    animal->ACBWT=(animal->ACBWT*(1+(.033*.5)));
    animal->MHETBWT=(animal->BWT*(.033*.5));
    animal->MHETADG=(animal->PADG*(.08534*.5));
}
else if ((animal->GSIREBREED==animal->GDAMBREED) && (animal->GSIREBREED!=animal->GGSIREBREED) && (animal->GSIREBREED!=animal->GGDAMBREED))

animal->MHETADG=(animal->PADG*(.08534*.5));
}
}
if (animal->Fgeneration==1){
if (animal->SIREBREED!=animal->DAMBREED){
animal->ACBWT=(animal->ACBWT*(1+.054));
animal->DHETBWT=((animal->BWT*.054) );
animal->DHETADG=((animal->PADG*.046) );
animal->DHETMWT=(animal->PMWT*.06);
}
else if (animal->Fgeneration==2){
if ((animal->SIREBREED!=animal->DAMBREED) && (animal->SIREBREED!=animal->GSIREBREED) && (animal->SIREBREED!=animal->GDAMBREED)){
animal->ACBWT=(animal->ACBWT*(1+.054));
animal->DHETBWT=((animal->BWT*.054) );
animal->DHETADG=((animal->PADG*.046) );
animal->DHETMWT=(animal->PMWT*.06);
}
else if ((animal->SIREBREED!=animal->DAMBREED) && (animal->SIREBREED==animal->GSIREBREED) || (animal->SIREBREED==animal->GDAMBREED)){
animal->ACBWT=(animal->ACBWT*(1+.054*.5));
animal->DHETBWT=((animal->BWT*.054*.5) );
animal->DHETADG=((animal->PADG*.046*.5) );
animal->DHETMWT=(animal->PMWT*((.06*.5)));
}
else if ((animal->SIREBREED==animal->DAMBREED) && (animal->SIREBREED!=animal->GSIREBREED) || (animal->SIREBREED!=animal->GDAMBREED)){
animal->ACBWT=(animal->ACBWT*(1+.054*.5));
animal->DHETBWT=((animal->BWT*.054*.5) );
animal->DHETADG=((animal->PADG*.046*.5) );
animal->DHETMWT=(animal->PMWT*((.06*.5)));
}
else if (animal->Fgeneration==3){
if ((animal->SIREBREED!=animal->DAMBREED) && (animal->SIREBREED!=animal->GSIREBREED) && (animal->SIREBREED!=animal->GDAMBREED) && (animal->SIREBREED!=animal->GGSIREBREED) && (animal->SIREBREED!=animal->GGDAMBREED)){
animal->ACBWT=(animal->ACBWT*(1+.054));
animal->DHETBWT=((animal->BWT*.054) );
animal->DHETADG=((animal->PADG*.046) );
animal->DHETMWT=(animal->PMWT*.06);
}
else if ((animal->SIREBREED!=animal->DAMBREED) && (animal->SIREBREED!=animal->GSIREBREED) && (animal->SIREBREED!=animal->GDAMBREED) && (animal->SIREBREED!=animal->GGSIREBREED) && (animal->SIREBREED!=animal->GGDAMBREED)){
animal->ACBWT=(animal->ACBWT*(1+.054));
animal->DHETBWT=((animal->BWT*.054) );
animal->DHETADG=((animal->PADG*.046) );
animal->DHETMWT=(animal->PMWT*.06);
}
>SIREBREED==animal->GGSIREBREED) || (animal->
>SIREBREED==animal->GGDAMBREED))
animal->ACBWT=(animal->ACBWT*(1+(.054*.75)));
animal->DHETBWT=(animal->BWT*((.054 *.75)))
animal->DHETADG=(animal->PADG*((.046 * .75)));
animal->DHETMWT=(animal->PMWT*((.06 * .75)));
}
else if ((animal->SIREBREED!=animal->DAMBREED) &&
(animal->SIREBREED==animal->GSIREBREED) || (animal->
SIREBREED==animal->GDAMBREED))
animal->ACBWT=(animal->ACBWT*(1+(.054*.5)));
animal->DHETBWT=(animal->BWT*((.054*.5)));
animal->DHETADG=(animal->PADG*((.046*.5)));
animal->DHETMWT=(animal->PMWT*((.06*.5)));
}
else if ((animal->SIREBREED==animal->DAMBREED) &&
(animal->SIREBREED!=animal->GSIREBREED) || (animal->
SIREBREED!=animal->GDAMBREED))
animal->ACBWT=(animal->ACBWT*(1+(.054*.5)));
animal->DHETBWT=(animal->BWT*((.054*.5)));
animal->DHETADG=(animal->PADG*((.046*.5)));
animal->DHETMWT=(animal->PMWT*((.06*.5)));
}
else if (animal->Fgeneration>3)
if ((animal->SIREBREED!=animal->DAMBREED) &&
(animal->SIREBREED!=animal->GSIREBREED) && (animal->
SIREBREED!=animal->GDAMBREED) && (animal->
SIREBREED!=animal->GGSIREBREED) && (animal->
SIREBREED!=animal->GGDAMBREED) && (animal->
SIREBREED!=animal->GGGSIREBREED) && (animal->
SIREBREED!=animal->GGGDAMBREED))
animal->ACBWT=(animal->ACBWT*(1+.054));
animal->DHETBWT=((animal->BWT*.054) );
animal->DHETADG=((animal->PADG*.046));
animal->DHETMWT=(animal->PMWT*.06);
}
else if ((animal->SIREBREED!=animal->DAMBREED) &&
(animal->SIREBREED!=animal->GSIREBREED) && (animal->
SIREBREED!=animal->GDAMBREED) && (animal->
SIREBREED!=animal->GGSIREBREED) && (animal->
SIREBREED!=animal->GGDAMBREED) && (animal->
SIREBREED==animal->GGGSIREBREED) || (animal->
SIREBREED==animal->GGGDAMBREED))
animal->ACBWT=(animal->ACBWT*(1+.054*.875));
animal->DHETBWT=(animal->BWT*((.054 * .875)));
animal->DHETADG=(animal->PADG*((.046 *.875)));
animal->DHETMWT=(animal->PMWT*((.06 * .875)));
}
else if ((animal->SIREBREED==animal->DAMBREED) &&
(animal->SIREBREED==animal->GSIREBREED) && (animal->
SIREBREED==animal->GGDAMBREED))
{418
>SIREBREED!=animal->GDAMBREED) && ((animal-
>SIREBREED==animal->GGSIREBREED) || (animal-
>SIREBREED==animal->GGDAMBREED)))
animal->ACBWT=(animal->ACBWT*(1+(.054*.75)));
animal->DHETBWT=(animal->BWT*((.054 * .75)));
animal->DHETADG=(animal->PADG*((.046 * .75)));
animal->DHETMWT=(animal->PMWT*((.06 * .75)));
}
else if ((animal->SIREBREED!=animal->DAMBREED) && ((animal-
>SIREBREED==animal->GSIREBREED) || (animal-
>SIREBREED==animal->GDAMBREED)))
animal->ACBWT=(animal->ACBWT*(1+(.054*.5)));
animal->DHETBWT=(animal->BWT*((.054*.5)));
animal->DHETADG=(animal->PADG*((.046*.5)));
animal->DHETMWT=(animal->PMWT*((.06*.5)));
}
else if ((animal->SIREBREED==animal->DAMBREED) && ((animal-
>SIREBREED!=animal->GSIREBREED) || (animal-
>SIREBREED!=animal->GDAMBREED)))
animal->ACBWT=(animal->ACBWT*(1+(.054*.5)));
animal->DHETBWT=(animal->BWT*((.054*.5)));
animal->DHETADG=(animal->PADG*((.046*.5)));
animal->DHETMWT=(animal->PMWT*((.06*.5)));
}

animal->BWT=( (animal->BWT) + (animal->DHETBWT) + (animal->
MHETBWT) );
aminal->CDWT=animal->BWT;
aminal->PADG=( (animal->PADG) + (animal->DHETADG) + (animal->
MHETADG) );
aminal->PMWT=( (animal->PMWT) + (animal->DHETMWT) );
if (animal->SEX==0){
mother->CBWT = animal->BWT;
aminal->CDWT=animal->CDWT *.453599;
aminal->SBWT=animal->CDWT *.96;
aminal->EBWT=animal->SBWT *.891;
aminal->EQSBW=((animal->SBWT) * (478/((animal->PMWT)
*.453599)));
aminal->EQEBW=animal->EQSBW *.891;
}
else if (animal->SEX==2){
mother->CBWT = animal->BWT;
aminal->CDWT=animal->CDWT *.453599;
aminal->SBWT=animal->CDWT *.96;
aminal->EBWT=animal->SBWT *.891;
aminal->EQSBW=((animal->SBWT) * (478/((animal->PMWT)
*.453599)));
aminal->EQEBW=animal->EQSBW *.891;
}
AddAnimal(animal);
void Cow::AddAnimal (AnimalRecord *animal){
    AnimalRecord *head;
    if (animal->ID == 1){
        herdhead=new AnimalRecord;
        head=herdhead;
        head->prev=NULL;
        head->next=NULL;
    } else{
        head = herdhead;
        while (head->next != NULL)
            head=head->next;
        head->next = new AnimalRecord;
        head->next->prev = head;
        head=head->next;
        head->next = NULL;
        if (animal->ID==0)
            animal->ID = head->prev->ID + 1;
    }
    head->ACBWT=animal->ACBWT;
    head->AGE=animal->AGE;
    head->AGED=animal->AGED;
    head->BCS=animal->BCS;
    head->BIRTHDATE=animal->BIRTHDATE;
    head->BREED=animal->BREED;
    head->BWT=animal->BWT;
    head->CADAT=animal->CADAT;
    head->CALFSEAS=animal->CALFSEAS;
    head->CBWT=animal->CBWT;
    head->CDWT=animal->CDWT;
    head->CDWTK=animal->CDWTK;
    head->CIDN=animal->CIDN;
    head->CBIRTHDATE=animal->CBIRTHDATE;
    head->CDWT=animal->CDWT;
    head->CNDAT=animal->CNDAT;
    head->COLR=animal->COLR;
    head->CSEX=animal->CSEX;
    head->CUCYC=animal->CUCYC;
    head->CULL=animal->CULL;
    head->DAMBREED=animal->DAMBREED;
    head->DETEC=animal->DETEC;
    head->DIE=animal->DIE;
    head->DPREG=animal->DPREG;
    head->DYSTO=animal->DYSTO;
    head->EBWTK=animal->EBWTK;
    head->EQSBW=animal->EQSBW;
    head->EQEBW=animal->EQEBW;
    head->GDAMBREED=animal->GDAMBREED;
    head->GGDAMBREED=animal->GGDAMBREED;
    head->GGGDAMBREED=animal->GGGDAMBREED;
}
head->GSIREBREED=animal->GSIREBREED;
head->GGSIREBREED=animal->GGSIREBREED;
head->GGGSIREBREED=animal->GGGSIREBREED;
head->GUWTK=animal->GUWTK;
head->FAIL=animal->FAIL;
head->Fgeneration=animal->Fgeneration;
head->FIDN=animal->FIDN;
head->FWEAN=animal->FWEAN;
head->ID=animal->ID;
head->IMPLANT=animal->IMPLANT;
head->LACT=animal->LACT;
head->LDAY=animal->LDAY;
head->MCMFC=animal->MCMFC;
head->MCSNF=animal->MCSNF;
head->MIDN=animal->MIDN;
head->PADG=animal->PADG;
head->UADG=animal->UADG;
head->UADG1=animal->UADG1;
head->UADG2=animal->UADG2;
head->UADG3=animal->UADG3;
head->UADG4=animal->UADG4;
head->PARIE=animal->PARIE;
head->PAST=animal->PAST;
head->PGAVG=animal->PGAVG;
head->PGEST=animal->PGEST;
head->PGSD=animal->PGSD;
head->PMSWT=animal->PMSWT;
head->PMWT=animal->PMWT;
head->PPI=animal->PPI;
head->PREG=animal->PREG;
head->PUBER=animal->PUBER;
head->INELIG=animal->INELIG;
head->Puberty=animal->Puberty;
head->PURE=animal->PURE;
head->REPLA=animal->REPLA;
head->SBWTK=animal->SBWTK;
head->WEAN=animal->WEAN;
head->WEANDATE=animal->WEANDATE;
head->WEANWT=animal->WEANWT;
head->DHETBWT=animal->DHETBWT;
head->DHETAGD=animal->DHETAGD;
head->DHETMWT=animal->DHETMWT;
head->MHETADG=animal->MHETADG;
head->MHETBWT=animal->MHETBWT;
head->SEAOP=animal->SEAOP;
head->SEAOP1=animal->SEAOP1;
head->SEAOP2=animal->SEAOP2;
head->SellCalf=animal->SellCalf;
head->SERVI=animal->SERVI;
head->SEX=animal->SEX;
head->SIREBREED=animal->SIREBREED;
head->SOUR=animal->SOUR;
head->UNPREG=animal->UNPREG;
head->YN=animal->YN;
head->YEN=animal->YEN;
head->next=NULL;
}

void Cow::DumpList(ofstream &test){
    /*A function that is used for the debugging & validation processes.*/
    AnimalRecord *head;
    head = herdhead;
    test << "------------------------------------------
    test << "Year: " << YEAR << " Date: " << currentDATE << endl;
    test << "------------------------------------------
    test << "Number of Breeding Cows: " <<BreedingCows
    test << "Number of Pregnant Cows: " <<PNUM<<endl;
    head=head->next;
    while  (head != NULL) {
        if ( ( head->SEX==3) || (head->REPLA==1) ){
            test << "Animal: " << head->ID << endl;
            test << "  Age: " << head->AGE << endl;
            test << "  Age of Death: " << head->AGED <<endl;
            test << "  Birthday: " << head->BIRTHDATE <<endl;
            test << "  Body Condition Score: " << (int)head->BCS << endl;
            test << "  Breed: " << (int)head->BREED << endl;
            test << "  Birth Weight: " << head->BWT << endl;
            test << "  BirthSeason: " << head->CALFSEAS<< endl;
            test << "  Calving Date: " << head->CADAT << endl;
            test << "  Calf Birth Weight: " << head->CBWT << endl;
            test << "  Current Daily Weight: " << head->CDWT << endl;
            test << "  Current Daily Weight in Kg: " << head->CDWTK << endl;
            test << "  Calf's ID Number: " << head->CIDN << endl;
            test << "  Calf's Date of Birth:  " << head->CBIRTHDATE <<endl;
            test << "  Conception Date: " << head->CNDAT << endl;
            test << "  Color: " << (int)head->COLR << endl;
            test << "  Calf Sex: " << (int) head->CSEX << endl;
            test << "  Current Cycle: " << head->CUCYC << endl;
            test << "  Cull: " << (int)head->CULL << endl;
            test << "  Detect Estrus: " << (int)head->DETEC << endl;
            test << "  Dead: " << (int)head->DIE << endl;
            test << "  Days Pregnant: " << (int)head->DPREG << endl;
        }
    }
}
Dystocia: " << (int)head->DYSTO << endl;
Empty Body Weight in Kg: " << head->EBWTK << endl;
Equivalent Shrunken Body Weight: " << head->EQSBW << endl;
Equivalent Empty Body Weight: " << head->EQEBW << endl;
Fail to Bear Calf: " << (int)head->FAIL << endl;
Fetus ID Number: " << (int)head->FIDN << endl;
Failed to Wean: " << (int)head->FWEAN << endl;
Gravid Uterus in Kg: " << head->GUWTK << endl;
Implant: " << (int)head->IMPLANT << endl;
Lactating: " << (int)head->LACT << endl;
Days Lactating: " << (double)head->LDAY << endl;
Milk Fat: " << head->MCMFC << endl;
Milk Solids: " << head->MCSNF << endl;
Mother ID Number: " << (int)head->MIDN << endl;
Potential Average Daily Gain: " << head->PADG << endl;
Parity: " << (int)head->PARIE << endl;
Animal on Pasture: " << (int)head->PAST << endl;
Length of Current Pregnancy: " << head->PGEST << endl;
Potential Mature Weight: " << head->PMWT << endl;
Post Partum Interval: " << head->PPI << endl;
Pregnant: " << (int)head->PREG << endl;
Date of Puberal Estrus: " << head->PUBER << endl;
Eligible or not to Breed: " << head->INELIG << endl;
Reached Puberty: " << (int)head->Puberty << endl;
PureBred or CrossBred: " << (int)head->PURE << endl;
Replacement Designated: " << (int)head->REPLA << endl;
Shrunken Body Weight in Kg: " << head->SBWTK << endl;
Weaning Weight in Lbs: " << head->WEANWT << endl;
Direct Heterosis in Lbs for ADG: " << head->DHETADG << endl;
Direct Heterosis for BirthWeight in Lbs: " << head->DHETBWT << endl;
Direct Heterosis in Lbs for Potential Mature Weight: " << head->DHETMWT << endl;
Maternal Heterosis in Lbs for ADG: " << head->MHETADG << endl;
Maternal Heterosis for BirthWeight in Lbs: " << head->MHETBWT << endl;
Number of Seasons Open: " << (int)head->SEAOP << endl;
Open after Spring Breeding Season: " << (int)head->SEAOP1 << endl;
test << " Open after Fall Breeding Season: " << (int)head->SEAOP2 << endl;
test << " Sell Orphan Calves: " << (int)head->SellCalf << endl;
test << " Estrus Periods Required: " << (int)head->SERVI << endl;
test << " Sex: " << (int)head->SEX << endl;
test << " Source: " << (int)head->SOUR << endl;
test << " Unable to conceive or maintain a pregnancy: " << (int)head->UNPREG << endl;
test << " Generation Number: " << (int)head->Fgeneration << endl;
test << " Sire Breed: " << (int)head->SIREBREED << endl;
test << " Grand Sire Breed: " << (int)head->GSIREBREED << endl;
test << " Great Grand Sire Breed: " << (int)head->GGSIREBREED << endl;
test << " Great Great Grand Sire Breed: " << (int)head->GGGSIREBREED << endl;
test << " Dam Breed: " << (int)head->DAMBREED << endl;
test << " Grand Dam Breed: " << (int)head->GDAMBREED << endl;
test << " Great Grand Breed: " << (int)head->GGDAMBREED << endl;
test << " Great Great Grand Breed: " << (int)head->GGGDAMBREED << endl;
test << " Potential Gestation Length(AVG): " << head->PGAVG << endl;
test << " Potential Gestation Length(SD): " << head->PGSD << endl;
test << " Potential Slaughter Weight: " << head->PMSWT << endl;
test << " Date Weaned: " << head->WEANDATE << endl;
}
head = head->next;
}
test << "----------------------------------------------------------" << endl;
}

void Cow::DeleteAnimal (int ID){
AnimalRecord *head;
AnimalRecord *tail;
AnimalRecord *mid;
head = GetAnimal(ID);
if (head == NULL){
    return;
}
if (head->next == NULL){
    if (head->prev == NULL){
        return;
    }
    tail = head;
    head = head->next;
    tail->prev = NULL;
    return;
}
else{
    mid = head->next;
    head->next = mid->next;
    mid->next = NULL;
    delete mid;
    return;
}
```cpp
void Cow::CalfMilk(AnimalRecord* animal) {
    AnimalRecord* tempcalf = NULL;
    if ((animal->SEX == 3) && (animal->CIDN > 0) && (animal->LACT == 1)) {
        if ((GetAnimal(animal->CIDN)) > 0) {
            tempcalf = GetAnimal(animal->CIDN);
            if ((tempcalf->CALFSEAS == 1) &&
                ((animal->LDAY + (tempcalf->BIRTHDATE)) < WEANSDATE) &&
                ((WEANSDATE - (animal->LDAY +
                (tempcalf->BIRTHDATE))) >= 0)) {
                tempcalf->YN = (animal->YN);
                tempcalf->YEN = (animal->YEN);
            }
        } else if ((tempcalf->CALFSEAS == 2) &&
                ((animal->LDAY + (tempcalf->BIRTHDATE)) < WEANFDATE) &&
                ((WEANFDATE - (animal->LDAY +
                (tempcalf->BIRTHDATE))) >= 0)) {
                tempcalf->YN = (animal->YN);
                tempcalf->YEN = (animal->YEN);
            }
        } else if ((tempcalf->CALFSEAS == 1) &&
                ((animal->LDAY + (tempcalf->BIRTHDATE)) < (WEANSDATE + 365)) &&
                ((WEANSDATE - (animal->LDAY +
                (tempcalf->BIRTHDATE))) < 0)) {
                tempcalf->YN = (animal->YN);
                tempcalf->YEN = (animal->YEN);
            }
        } else if ((tempcalf->CALFSEAS == 2) &&
                ((animal->LDAY + (tempcalf->BIRTHDATE)) < (WEANFDATE + 365)) &&
                ((WEANFDATE - (animal->LDAY +
                (tempcalf->BIRTHDATE))) < 0)) {
                tempcalf->YN = (animal->YN);
```
tempcalf->YEN = (animal->YEN);
}
else{
    animal->CIDN=0;
    animal->LACT=0;
    animal->LDAY=0;
    animal->CIDN=0;
    animal->CBIRTHDATE=0;
    animal->MCMFC=0;
    animal->MCSNF=0;
    animal->YN=0;
    animal->YEN=0;
}
}
else{
    return;
}
}

void Cow::DeleteHerd(){
    int bottom = LastID();
    int loop;
    for (loop = bottom; loop > 0; loop--){
        DeleteAnimal(loop);
    }
}

void Cow::CreateHerd(){
    int count=1;
    int TBRED[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    int j;
    int i;
    int k;
    int l;
    double BWTAVG;
    double BSD;
    double PADGAVG;
    double PADGSD;
    double PMTAVG;
    double PMSD;
    double PMSSD;
    double PMSTAVG;
    AnimalRecord *tempcow;
    TBRED[0]=0;
    TBRED[2]=COWS*PER[2];
    TBRED[3]=COWS*PER[3];
    TBRED[4]=COWS*PER[4];
    TBRED[5]=COWS*PER[5];
    TBRED[6]=COWS*PER[6];
TBRED[7] = COWS * PER[7];
TBRED[8] = COWS * PER[8];
TBRED[9] = COWS * PER[9];
TBRED[10] = COWS * PER[10];
TBRED[12] = COWS * PER[12];
TRACE("CreateHerd is Called.\n");
while (count <= CONUM){
  TRACE("COUNT=%i\n", count);
  tempcow = new AnimalRecord;
  tempcow->ID = count;
  if (PURE == 0){
    tempcow->DAMBREED = HBREED;
    tempcow->PURE = PURE;
    tempcow->SIREBREED = HBREED;
    tempcow->BREED = HBREED;
    switch (tempcow->SIREBREED){
      case 1:  tempcow->CDWT = GetStochasticNumber(139.96, 1156);
                tempcow->BWT = GetStochasticNumber(9.91, 80.9);
                tempcow->PADG = GetStochasticNumber(.16, 1.87);
                tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
                tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
                tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
                tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
                tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
                tempcow->PMSWT = GetStochasticNumber(26.0, 969);
                tempcow->PGAVG = 285.4;
                tempcow->PGSD = 6.02;
                tempcow->Fgeneration = 0; break;
      case 2:  tempcow->CDWT = GetStochasticNumber(143.84, 1188);
                tempcow->BWT = GetStochasticNumber(10.56, 86.2);
                tempcow->PADG = GetStochasticNumber(.145, 1.70);
                tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
                tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
                tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
                tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
                tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
                tempcow->PMSWT = GetStochasticNumber(27.85, 1038);
                tempcow->PGAVG = 288.6;
                tempcow->PGSD = 6.09;
                tempcow->Fgeneration = 0; break;
    }
  }
}

427
case 3:  tempcow->CDWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
tempcow->PADG=GetStochasticNumber(.145 , 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
tempcow->Fgeneration=0; break;
case 4:  tempcow->CDWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.51 , 85.8);
tempcow->PADG=GetStochasticNumber(.154 , 1.8);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.51 , 988);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
tempcow->Fgeneration=0; break;
case 5:  tempcow->CDWT=GetStochasticNumber(168.42, 1391);
tempcow->BWT =GetStochasticNumber(11.1 , 90.6);
tempcow->PADG=GetStochasticNumber(.158 , 1.85);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218 , 1238);
tempcow->PGAVG=285.9;
tempcow->PGSD=6.03;
tempcow->Fgeneration=0; break;
case 6:  tempcow->CDWT=GetStochasticNumber(189.24, 1563);
tempcow->BWT =GetStochasticNumber(10.94 , 89.3);
tempcow->PADG=GetStochasticNumber(.157 , 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(36.223 , 1350);
tempcow->PGAVG=287.5;
tempcow->PGSD=6.07;
tempcow->Fgeneration=0; break;
case 7:  tempcow->CDWT=GetStochasticNumber(156.67, 1294);
tempcow->BWT =GetStochasticNumber(10.54 , 86.0);
tempcow->PADG=GetStochasticNumber(.160 , 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(31.527 , 1175);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
tempcow->Fgeneration=0; break;
case 8:  tempcow->CDWT=GetStochasticNumber(146.62, 1211);
tempcow->BWT =GetStochasticNumber(10.51 , 85.8);
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(29.864 , 1113);
tempcow->PGAVG=288.1;
tempcow->PGSD=6.08;
tempcow->Fgeneration=0; break;
case 9:  tempcow->CDWT=GetStochasticNumber(184.16, 1521);
tempcow->BWT =GetStochasticNumber(11.1 , 90.6);
tempcow->PADG=GetStochasticNumber(.154 , 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218 , 1238);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
tempcow->Fgeneration=0; break;
case 10: tempcow->CDWT=GetStochasticNumber(149.17, 1232);
tempcow->BWT =GetStochasticNumber(10.58 , 86.4);
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844 , 1075);
tempcow->PGAVG=286.0;
tempcow->PGSD=6.03;
tempcow->Fgeneration=0; break;
case 11: tempcow->CDWT=GetStochasticNumber(156.91, 1296);
tempcow->BWT =GetStochasticNumber(10.89 , 88.9);
tempcow->PADG=GetStochasticNumber(.157 , 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(32.547 , 1213);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
tempcow->Fgeneration=0; break;
case 12: tempcow->CDWT=GetStochasticNumber(137.3, 1134);
tempcow->BWT=GetStochasticNumber(10.13, 82.7);
tempcow->PADG=GetStochasticNumber(.154, 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.495, 1062);
tempcow->PGAVG=287.1;
tempcow->PGSD=6.06;
tempcow->Fgeneration=0; break;
}
else if (PURE != 0){
  j=0;
  k=0;
  l=0;
tempcow->BREED=0;
tempcow->SIREBREED=0;
tempcow->DAMBREED=0;
tempcow->GSIREBREED=0;
tempcow->GDAMBREED=0;
      (PER[1] >= PER[11]) && (PER[1] >= PER[12]) && (tempcow->SIREBREED==0)) {
    k=1;
    tempcow->BREED=1;
    tempcow->SIREBREED=1;
  }
    k=2;
    tempcow->BREED=2;
    tempcow->SIREBREED=2;
}
           (PER[3] >= PER[4])
           PER[7])
           PER[10])
           SIREBREED==0) ){
    k=3;
tempcow->BREED=3;
tempcow->SIREBREED=3;
}
           (PER[4] >= PER[3]) &&
           PER[7]) &&
           PER[11]) &&
           (PER[4] >= PER[12]) && (tempcow->SIREBREED==0) ){
    k=4;
tempcow->BREED=4;
tempcow->SIREBREED=4;
}
           (PER[5] >= PER[3]) &&
           PER[7]) &&
           PER[10]) &&
           SIREBREED==0) ){
    k=5;
tempcow->BREED=5;
tempcow->SIREBREED=5;
}
           (PER[6] >= PER[3]) &&
           PER[7]) &&
           PER[10]) &&
           SIREBREED==0) ){
    k=6;
tempcow->BREED=6;
tempcow->SIREBREED=6;
}
PER[8]) && (PER[7] >= PER[9]) &&
PER[12]) && (tempcow->SIREBREED==0) ){ 
k=7;
tempcow->BREED=7;
tempcow->SIREBREED=7;
}
else if ( (PER[8] >= PER[1]) && (PER[8] >= PER[2]) &&
(PER[8] >= PER[3]) && (PER[8] >= PER[4]) &&
PER[11]) && (PER[8] >= PER[12]) && (tempcow->SIREBREED==0) ){ 
k=8;
tempcow->BREED=8;
tempcow->SIREBREED=8;
}
k=9;
tempcow->BREED=9;
tempcow->SIREBREED=9;
}
else if ( (PER[10] >= PER[1]) && (PER[10] >= PER[2]) &&
PER[7]) && (PER[10] >= PER[8]) && (PER[10] >= PER[9]) &&
PER[12]) && (tempcow->SIREBREED==0) ){ 
k=10;
tempcow->BREED=10;
tempcow->SIREBREED=10;
}
PER[12]) && (tempcow->SIREBREED==0) ){ 
k=11;
tempcow->BREED=11;
tempcow->SIREBREED=11;
}
else if ( (PER[12] >= PER[1]) && (PER[12] >= PER[2]) &&
PER[7]) && (PER[12] >= PER[8]) &&
{
    k=12;
tempcow->BREED=12;
tempcow->SIREBREED=12;
}
for (i=1; i<13; i++){
    if ((PER[i]>=PER[i-1]) && (PER[i] <=PER[k]) && (i != tempcow->SIREBREED) && (PER[i] !=0))
    {
        j=i;
tempcow->DAMBREED=i;
    }
}
for (i=1; i<13; i++){
    if ((PER[i]>=PER[i-1]) && (PER[i] <=PER[j]) && (PER[i] <=PER[k]) && (i != tempcow->SIREBREED) && (i != tempcow->DAMBREED) && (PER[i] !=0))
    {
        l=i;
tempcow->GSIREBREED=i;
    }
}
for (i=1; i<13; i++){
    if ((PER[i]>=PER[i-1]) && (PER[i] <=PER[j]) && (PER[i] <=PER[k]) && (i != tempcow->SIREBREED) && (i != tempcow->DAMBREED) && (i != tempcow->GSIREBREED) && (PER[i] !=0))
    {
        tempcow->GDAMBREED=i;
    }
}
if ((tempcow->SIREBREED>0) && (tempcow->DAMBREED==0))
{
    tempcow->DAMBREED=tempcow->SIREBREED;
}
BWTAVG =
((80.9*PER[1])+(86.2*PER[2])+(86.2*PER[3])+(85.8*PER[4])+(90 .6*PER[5])
+(89.3*PER[6])+(86.0*PER[7])+(85.8*PER[8])+(90.6*PER[9])+(86 
.4*PER[10])
+(88.9*PER[11])+(82.7*PER[12]));
BSD=(.1225*BWTAVG);
tempcow->BWT=GetStochasticNumber(BSD , BWTAVG);
tempcow->ACBWT=BWTAVG;
PMTAVG =
((1156*PER[1])+(1188*PER[2])+(1188*PER[3])+(1188*PER[4])+(13 
91*PER[5])
+(1563*PER[6])+(1294*PER[7])+(1211*PER[8])+(1521*PER[9])+(12 
32*PER[10])
+(1296*PER[11])+(1134*PER[12]));
PMSD = (.12105*PMTAVG);
tempcow->CDWT = GetStochasticNumber(PMSD, PMTAVG);
PADGAVG =
((1.87*PER[1])+(1.70*PER[2])+(1.70*PER[3])+(1.80*PER[4])+(1.
85*PER[5])
  + (1.83*PER[6]) + (1.87*PER[7]) + (1.76*PER[8]) + (1.81*PER[9]) + (1.76*PER[10])
  + (1.83*PER[11]) + (1.81*PER[12]));
PADGSD  = (.11312*PADGAVG);
tempcow->PADG=GetStochasticNumber(PADGSD, PADGAVG);
tempcow->PGAVG  =
  ((285.4*PER[1]) + (288.6*PER[2]) + (288.6*PER[3]) + (288.6*PER[4])
  + (285.9*PER[5])
  + (287.5*PER[6]) + (286.3*PER[7]) + (288.1*PER[8]) + (285.4*PER[9])
  + (286.0*PER[10])
  + (286.3*PER[11]) + (287.1*PER[12]));
tempcow->PGSD  = (.02110* tempcow->PGAVG);
PMSTAVG
  =(((969*PER[1]) + (1038*PER[2]) + (1038*PER[3]) + (988*PER[4]) + (1238*PER[5])
  + (1350*PER[6]) + (1175*PER[7]) + (1113*PER[8]) + (1238*PER[9]) + (1075*PER[10])
  + (1213*PER[11]) + (1062*PER[12]));
PMSSD  = (.02683*PMSTAVG);
tempcow->PMSWT=GetStochasticNumber(PMSSD , PMSTAVG);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->Fgeneration+= 1;
}
tempcow->AGE= (int) GetStochasticNumber(275, 1730);
tempcow->AGED=0;
tempcow->BCS=5;
tempcow->BIRTHDATE=0;
tempcow->CADAT=0;
tempcow->CALFSEAS=0;
tempcow->CBWT=0;
tempcow->CDWTK=(tempcow->CDWT*.453599);
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->CNDAT=0;
tempcow->COLR=COLR;
tempcow->CSEX=0;
tempcow->CUCYC=0;
tempcow->CULL=0;
tempcow->DETEC=0;
tempcow->DIE=0;
tempcow->DPREG=0;
tempcow->DYSTO=0;
tempcow->EBWTK=0;
tempcow->EQSBW=0;
tempcow->EQEBW=0;
tempcow->FAIL=0;
tempcow->FIDN=0;
tempcow->FWEAN=0;
tempcow->GGGDAMBREED=0;
tempcow->GGGSIREBREED=0;
tempcow->GGDAMBREED=0;
tempcow->GGSIREBREED=0;
tempcow->GUWTK=0;
tempcow->IMPLANT=0;
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->MIDN=0;
if (tempcow->AGE <= 730) {
    tempcow->PARIE= 0;
}
else if ( (tempcow->AGE > 730) && (tempcow->AGE <= 1096) ) { 
    tempcow->PARIE= 1;
}
else if (tempcow->AGE > 1096) {
    tempcow->PARIE=2;
}
tempcow->PAST=1;
tempcow->PGEST=0;
tempcow->PMWT=tempcow->CDWT;
tempcow->PPI=0;
tempcow->PREG=0;
tempcow->PUBER=0;
tempcow->PURE=PURE;
tempcow->INELIG=0;
tempcow->Puberty=1;
tempcow->REPLA=0;
tempcow->SBWTK=((tempcow->CDWT*.453599)* .96);
tempcow->WEAN=0;
tempcow->WEANWT=0;
tempcow->WEANDATE=0;
tempcow->DHETBWT=0;
tempcow->DHETADG=0;
tempcow->DHETMWT=0;
tempcow->MHETADG=0;
tempcow->MHETBWT=0;
tempcow->SEAOP=0;
tempcow->SEAOP1=0;
tempcow->SEAOP2=0;
tempcow->SellCalf=0;
tempcow->SERVI=0;
```c
    tempcow->SEX=3;
    tempcow->SOUR=0;
    tempcow->UNPREG=0;
    tempcow->YN=0;
    tempcow->YEN=0;
    tempcow->next=NULL;
    AddAnimal(tempcow);
    count++;
    delete tempcow;
}
TRACE("Finished with herd.\n");
}

int Cow::LastID(){
    AnimalRecord *head;
    head = herdhead;
    while (head->next != NULL)
        head=head->next;
    return head->ID;
}
AnimalRecord* Cow::GetAnimal(int ID){
    AnimalRecord *head;
    head=herdhead;
    while (head != NULL){
        if (head->ID == ID){
            return head;
        }
        head = head->next;
    }
    return NULL;
}
AnimalRecord* Cow::GetHeadAnimal(){
    return herdhead;
}
void Cow::Lactation(AnimalRecord *animal){
    if ( (animal->CIDN>0) && (animal->LACT==1) && ((animal->LDAY - 1) + animal->CBIRTHDATE)>WEANSDATE) && ((animal->CBIRTHDATE) < 182) && (WEANSDATE - (animal->LDAY + animal->CBIRTHDATE) >= 0)){
        animal->CIDN=0;
        animal->CBIRTHDATE=0;
        animal->LDAY=0;
        animal->LACT=0;
        animal->MCMFC=0;
        animal->MCSNF=0;
        animal->YN=0;
        animal->YEN=0;
    }
```
else if ( (animal->CIDN>0) && (animal->LACT==1) && ((animal->LDAY - 1) + animal->CBIRTHDATE)>(WEANSDATE + 365) ) && ((animal->CBIRTHDATE) < 182) && ((WEANSDATE - (animal->LDAY + animal->CBIRTHDATE)) < 0)){
    animal->CIDN=0;
    animal->CBIRTHDATE=0;
    animal->LDAY=0;
    animal->LACT=0;
    animal->MCMFC=0;
    animal->MCSNF=0;
    animal->YN=0;
    animal->YEN=0;
}
else if ( (animal->CIDN>0) && (animal->LACT==1) && (((animal->LDAY - 1) + animal->CBIRTHDATE)>WEANFDATE) && (animal->CBIRTHDATE) >= 182) && (WEANFDATE - (animal->LDAY + animal->CBIRTHDATE) >= 0)){
    animal->CIDN=0;
    animal->CBIRTHDATE=0;
    animal->LDAY=0;
    animal->LACT=0;
    animal->MCMFC=0;
    animal->MCSNF=0;
    animal->YN=0;
    animal->YEN=0;
}
else if ( (animal->CIDN>0) && (animal->LACT==1) && (((animal->LDAY - 1) + animal->CBIRTHDATE)>(WEANFDATE + 365) ) && ((animal->CBIRTHDATE) >= 182) && ((WEANFDATE - (animal->LDAY + animal->CBIRTHDATE)) < 0)){
    animal->CIDN=0;
    animal->CBIRTHDATE=0;
    animal->LDAY=0;
    animal->LACT=0;
    animal->MCMFC=0;
    animal->MCSNF=0;
    animal->YN=0;
    animal->YEN=0;
}
}

void Cow::ChangeSex(AnimalRecord *animal){
    AnimalRecord *tempcow;
    if ( (animal->CALFSEAS==1) && (currentDATE==WEANSDATE) ){
        if (((animal->AGE - 1) + (animal->BIRTHDATE)) == WEANSDATE) && (((animal->AGE - 1) + (animal->BIRTHDATE)) < 365) ){
            if ((animal->SEX < 3) && (animal->SellCalf==0)){
                if ( (animal->SEX == 0) && (animal->REPLA == 0) ){
                    animal->SEX = 4;
                    animal->WEANWT=animal->CDWT;
                }
            }
        }
    }
}
if (animal->WEAN==0) {
    TWHS += animal->CDWT;
    NHFC++;
    TNCW++;
    WNWT += animal->CDWT;
    animal->YN=0;
    animal->YEN=0;
    animal->WEAN=animal->AGE;
    TSWEANAGE+=animal->WEAN;
    SCALF++;
}
if (animal->WEANDATE==0) {
    animal->WEANDATE=currentDATE;
}
animal->PADG= ( (animal->PADG) - (animal->MHETADG) );
TNCOH--;
TNSCOH++;
if (animal->MIDN != 0) {
    tempcow = GetAnimal(animal->MIDN);
    if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
        (tempcow->CIDN==animal->ID) ) ) {
        tempcow->CIDN=0;
        tempcow->CBIRTHDATE=0;
        tempcow->LDAY=0;
        tempcow->LACT=0;
        tempcow->MCMFC=0;
        tempcow->MCSNF=0;
        tempcow->YN=0;
        tempcow->YEN=0;
    }
}
else if (animal->SEX == 2) {
    animal->SEX = 5;
    animal->WEANWT=animal->CDWT;
    animal->PADG= ( (animal->PADG) - (animal->MHETADG) );
    if (animal->WEAN==0) {
        TWSS += animal->CDWT;
        NSTC++;
        TNCW++;
        WNWT += animal->CDWT;
        animal->YN=0;
        animal->YEN=0;
        animal->WEAN=animal->AGE;
        TSWEANAGE+=animal->WEAN;
        SCALF++;
    }
    if (animal->WEANDATE==0) {
        animal->WEANDATE=currentDATE;
    }
    TNCOH--;
}
TNSCOH++;
if (animal->MIDN != 0){
tempcow = GetAnimal(animal->MIDN);
if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
(tempcow->CIDN==animal->ID) ) ){
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->LDAY=0;
tempcow->LACT=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
}
}
else if ( (animal->SEX == 0) && (animal->REPLA==1) ){
animal->SEX=3;
animal->WEANWT=animal->CDWT;
if (animal->WEAN==0){
  REPLF ++;
  TNCW++;
  WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
animal->WEAN=animal->AGE;
  TSWEANAGE+=animal->WEAN;
  SCALF++;
}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
animal->PADG=( (animal->PADG) - (animal->MHETADG) );
TNSCOH--;
BreedingCows++; 
BreedingReplacement--; 
if (animal->MIDN != 0){
tempcow = GetAnimal(animal->MIDN);
if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
(tempcow->CIDN==animal->ID) ) ){
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->LDAY=0;
tempcow->LACT=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
}
}
else if ( (((animal->AGE - 1) + (animal->BIRTHDATE)) == (WEANSDATE + 365)) && (((animal->AGE - 1) + (animal->BIRTHDATE)) > 365) ){  
if ((animal->SEX < 3) && (animal->SellCalf==0)){
  if ( (animal->SEX == 0) && (animal->REPLA == 0) ){ 
    animal->SEX = 4;
    animal->WEANWT=animal->CDWT;
    if (animal->WEAN==0){
      TWHS += animal->CDWT;
      NHFC++;
      TNCW++;
      WNWT += animal->CDWT;
      animal->YN=0;
      animal->YEN=0;
      animal->WEAN=animal->AGE;
      TSWEANAGE+=animal->WEAN;
      SCALF++;
    }
    if (animal->WEANDATE==0){
      animal->WEANDATE=currentDATE;
    }
    animal->PADG= ( (animal->PADG) - (animal->MHETADG) );
    TNCOH--;
    TNSCOH++;
    if (animal->MIDN != 0){
      tempcow = GetAnimal(animal->MIDN);
      if ( (tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
                     (tempcow->CIDN==animal->ID) ) ){ 
        tempcow->CIDN=0;
        tempcow->CBIRTHDATE=0;
        tempcow->LDAY=0;
        tempcow->LACT=0;
        tempcow->MCMFC=0;
        tempcow->MCSNF=0;
        tempcow->YN=0;
        tempcow->YEN=0;
      }
    }
  }
}
else if (animal->SEX == 2){
  animal->SEX = 5;
  animal->WEANWT=animal->CDWT;
  if (animal->WEAN==0){
    TWSS += animal->CDWT;
    NSTC++;
    TNCW++;
    WNWT += animal->CDWT;
    animal->YN=0;
    animal->YEN=0;
  }
}
animal->WEAN=animal->AGE;
TSWEANAGE+=animal->WEAN;
SCALF++;
}
if (animal->WEANDATE==0){
    animal->WEANDATE=currentDATE;
}
animal->PADG=( (animal->PADG) - (animal->MHETADG) );
TNCOH--;
TNSCOH++;
if (animal->MIDN != 0){
tempcow = GetAnimal(animal->MIDN);
if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
( tempcow->CIDN==animal->ID) ) ){
    tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->LDAY=0;
tempcow->LACT=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
}
}
else if ( (animal->SEX == 0) && (animal->REPLA==1) ){
    animal->SEX=3;
animal->WEANWT=animal->CDWT;
if (animal->WEAN==0){
    REPLF++;
    NSTC++;
    TNCW++;
    WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
animal->WEAN=animal->AGE;
TSWEANAGE+=animal->WEAN;
SCALF++;
}
if (animal->WEANDATE==0){
    animal->WEANDATE=currentDATE;
}
animal->PADG=( (animal->PADG) - (animal->MHETADG) );
TNCOH--;
BreedingCows++;
BreedingReplacement--; 
if (animal->MIDN != 0){
tempcow = GetAnimal(animal->MIDN);
if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
( tempcow->CIDN==animal->ID) ) ){
    tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->LDAI=0;
tempcow->LACT=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
}
}
}
}
}
}
else if ( (animal->CalfSEAS==2) && (currentDATE==WEANFDATE) )
{
  if ( (((animal->AGE - 1) + (animal->BIRTHDATE)) == WEANFDATE) && (((animal->AGE - 1) + (animal->BIRTHDATE))<=365) )
  {
    if ( (animal->SEX < 3) && (animal->SellCalf==0) )
      if ( (animal->SEX == 0) && (animal->REPLA == 0) )
        animal->SEX = 4;
    animal->WEANWT=animal->CDWT;
    if (animal->WEAN==0)
      TWHS += animal->CDWT;
    NHFC++;
    TNCW++;
    WNWT += animal->CDWT;
    animal->YN=0;
    animal->YEN=0;
    FCALF++;
    animal->WEAN=animal->AGE;
    TFWEANAGE+=animal->WEAN;
  }
  if (animal->WEANDATE==0)
    animal->WEANDATE=currentDATE;
}
animal->PADG=( (animal->PADG) - (animal->MHETADG) );
TNCOH--;
TNCOH++;
if (animal->MIDN != 0)
  tempcow = GetAnimal(animal->MIDN);
  if ( ( tempcow != NULL) && ( (tempcow->ID==animal->MIDN) &&
    (tempcow->CIDN==animal->ID) ) )
  {
    tempcow->CIDN=0;
    tempcow->CBIRTHDATE=0;
    tempcow->LDAI=0;
    tempcow->LACT=0;
    tempcow->MCMFC=0;
    tempcow->MCSNF=0;
    tempcow->YN=0;
    tempcow->YEN=0;
}
else if (animal->SEX == 2){
    animal->SEX = 5;
    animal->WEANWT=animal->CDWT;
    if (animal->WEAN==0){
        TWSS += animal->CDWT;
        NSTC++;
        TNCW++;
        WNWT += animal->CDWT;
        animal->YN=0;
        animal->YEN=0;
        FCALF++;
        animal->WEAN=animal->AGE;
        TFWEANAGE+=animal->WEAN;
    }
    if (animal->WEANDATE==0){
        animal->WEANDATE=currentDATE;
    }
    animal->PADG= ( (animal->PADG) - (animal->MHETADG) );
    TNCOH--;
    TNSCOH++;
    if (animal->MIDN != 0){
        tempcow = GetAnimal(animal->MIDN);
        if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
            (tempcow->CIDN==animal->ID) ) ){
            tempcow->CIDN=0;
            tempcow->CBIRTHDATE=0;
            tempcow->LDAY=0;
            tempcow->LACT=0;
            tempcow->MCMFC=0;
            tempcow->MCSNF=0;
            tempcow->YN=0;
            tempcow->YEN=0;
        }
    }
}
else if ( (animal->SEX == 0) && (animal->REPLA==1) ){
    animal->SEX=3;
    animal->WEANWT=animal->CDWT;
    if (animal->WEAN==0){
        REPLF++;
        NSTC++;
        TNCW++;
        WNWT += animal->CDWT;
        animal->YN=0;
        animal->YEN=0;
        FCALF++;
        animal->WEAN=animal->AGE;
TFWEANAGE+=animal->WEAN;
}  
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}  
animal->PADG= ( (animal->PADG) - (animal->MHETADG) );
TNCOH--;
BreedingCows++;
BreedingReplacement--;
if (animal->MIDN != 0){
tempcow = GetAnimal(animal->MIDN);
if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
( (tempcow->CIDN==animal->ID) ) ) ){
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->LDAY=0;
tempcow->LACT=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
}
}
}
}else if ( (((animal->AGE - 1) + (animal->BIRTHDATE)) ==
(WEANFDATE + 365) ) && (((animal->AGE - 1) + (animal->BIRTHDATE)) > 365) ){
if ((animal->SEX < 3) && (animal->SellCalf==0)){
if ( (animal->SEX == 0) && (animal->REPLA == 0) ){
animal->SEX = 4;
animal->WEANWT=animal->CDWT;
if (animal->WEAN==0){
TWHS += animal->CDWT;
NHFC++;
TNCW++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
FCALF++;
animal->WEAN=animal->AGE;
TFWEANAGE+=animal->WEAN;
}
}
}if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}  
animal->PADG= ( (animal->PADG) - (animal->MHETADG) );
TNCOH--;
TNSCOH++;  
if (animal->MIDN != 0){
tempcow = GetAnimal(animal->MIDN);
if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
(tempcow->CIDN==animal->ID) ) ){ 
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->LDAY=0;
tempcow->LACT=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
}
}
else if (animal->SEX == 2){
animal->SEX = 5;
animal->WEANWT=animal->CDWT;
if (animal->WEAN==0){
TWSS += animal->CDWT;
NSTC++;
TNCW++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
FCALF++;
animal->WEAN=animal->AGE;
TFWEANAGE+=animal->WEAN;
}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
animal->PADG= ( (animal->PADG) - (animal->MHETADG) );
TNCOH--;
TNSCOH++;
if (animal->MIDN != 0){
tempcow = GetAnimal(animal->MIDN);
if ( ( tempcow !=NULL) && ( (tempcow->ID==animal->MIDN) &&
(tempcow->CIDN==animal->ID) ) ){ 
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->LDAY=0;
tempcow->LACT=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
}
}
else if ( (animal->SEX == 0) && (animal->REPLA==1) ){
animal->SEX=3;
animal->WEANWT=animal->CDWT;
if (animal->WEAN==0){
  REPLF++;
  NSTC++;
  TNCW++;
  WNWT += animal->CDWT;
  animal->YN=0;
  animal->YEN=0;
  FCALF++;
  animal->WEAN=animal->AGE;
  TFWEANAGE+=animal->WEAN;
}
if (animal->WEANDATE==0){
  animal->WEANDATE=currentDATE;
}
animal->PADG=((animal->PADG) - (animal->MHETADG));
TNCOH--;
BreedingCows++;
BreedingReplacement--;
if (animal->MIDN != 0){
  tempcow = GetAnimal(animal->MIDN);
  if ( (tempcow != NULL) && (tempcow->ID==animal->MIDN) &&
      (tempcow->CIDN==animal->ID) ){
    tempcow->CIDN=0;
    tempcow->CBIRTHDATE=0;
    tempcow->LDAY=0;
    tempcow->LACT=0;
    tempcow->MCMFC=0;
    tempcow->MCSNF=0;
    tempcow->YN=0;
    tempcow->YEN=0;
  }
}
void Cow::PregnancyTest(AnimalRecord *animal){
  if ( (TEST==0) && (animal->SEX==3) && (animal->Puberty==1)
      && (animal->INELIG==0) ){
    if ((CULPO==0) && (SpringCalfSeasn==1)){
      if ( (currentDATE==(LDBS + PTEST)) ){
        if (animal->PREG==0) {
          animal->SEAOP++;;
          EligCowsOpen++;
          TSTOP++;
        }
      }
    }
  }
}
else if ((CULPO==0) && (FallCalfSeasn==1)){
    if ( ((LDBS2 + PTEST)<=365) && (currentDATE==(LDBS2 + 
PTEST))){
        if ((animal->PREG==0) && (animal->INELIG == 0)){
            animal->SEAOP++;
            EligCowsOpen++;
            TSTOP++;
            if (animal->UNPREG==1){
                Sterile++;
            }
        }
    }
    else if ( (LDBS2 + PTEST>365) && (currentDATE==LDBS2 + PTEST 
- 365) && (LDBS2<=365)){
        if (animal->PREG==0) {
            animal->SEAOP++;
            EligCowsOpen++;
            TSTOP++;
            if (animal->UNPREG==1){
                Sterile++;
            }
        }
    }
    else if ( (LDBS2 + PTEST>365) && (currentDATE==LDBS2 + PTEST 
- 365) && (LDBS2>365)){
        if (animal->PREG==0){
            animal->SEAOP++;
            EligCowsOpen++;
            TSTOP++;
            if (animal->UNPREG==1){
                Sterile++;
            }
        }
    }
    else if (CULPO>0){
        if (NumBreedSeasns==1){
            if (SpringCalfSeasn==1){
                if ( (currentDATE==(LDBS + PTEST))){
                    if ((animal->PREG==0) && (animal->INELIG == 0)){
                        animal->SEAOP++;
                        EligCowsOpen++;
                        if (animal->UNPREG==1){
                            Sterile++;
                        }
                    }
                }
            }
        }
    }
}
else if (FallCalfSeasn==1){
    if ( (LDBS2 + PTEST<=365) && (currentDATE==LDBS2 + PTEST) ){
        if ((animal->PREG==0) && (animal->INELIG == 0)){
            animal->SEAOP++;  
            EligCowsOpen++;  
            if (animal->UNPREG==1){
                Sterile++;  
            }
        }
    }
}
else if ( (LDBS2 + PTEST>365) && (currentDATE==LDBS2 + PTEST - 365) &&
    (simulationDAY>(LDBS2 + PTEST-365)) && (LDBS2<=365) ){
    if ((animal->PREG==0) && (animal->INELIG == 0)){
        animal->SEAOP++;  
        EligCowsOpen++;  
        if (animal->UNPREG==1){
            Sterile++;  
        }
    }
}
else if ( (LDBS2 + PTEST>365) && (currentDATE==LDBS2 + PTEST - 365) &&
    (simulationDAY>(LDBS2 + PTEST-365)) && (LDBS2>365) ){
    if ((animal->PREG==0) && (animal->INELIG == 0)){
        animal->SEAOP++;  
        EligCowsOpen++;  
        if (animal->UNPREG==1){
            Sterile++;  
        }
    }
}
else if (NumBreedSeasns==2){
    if (SpringCalfSeasn==1){
        if ( (currentDATE==(LDBS + PTEST))){
            if ((animal->PREG==0) && (animal->INELIG == 0)){
                animal->SEAOP1=1;  
            }
        }
    }
    if (FallCalfSeasn==1){
        if ( ( ((LDBS2 + PTEST) <=365) && (currentDATE==((LDBS2 +
            PTEST))) ){
            if ((animal->PREG==0) && (animal->INELIG == 0)){
                animal->SEAOP2=1;  
            }
        }
    }
}
else if ( ( (LDBS2 + PTEST)>365) && (currentDATE==((LDBS2 +
    PTEST) - 365)) &&}
if ( simulationDAY>=(LDBS2 + PTEST)-365) && (LDBS2<=365) ){
    if ((animal->PREG==0) && (animal->INELIG == 0)){
        animal->SEAOP2=1;
    }
} else if ( (LDBS2 + PTEST)>365) && (currentDATE==((LDBS2 + PTEST) - 365)) &&
    (simulationDAY>=(LDBS2 + PTEST)-365) && (LDBS2>365) ){
    if ((animal->PREG==0) && (animal->INELIG == 0)){
        animal->SEAOP2=1;
    }

} else if ( ((LDBS2 + PTEST)<=365) && (currentDATE==365) &&
    ((animal->SEAOP1 + animal->SEAOP2)==2) ){
    animal->FAIL++;
    animal->SEAOP1=0;
    animal->SEAOP2=0;
    CowsFail++;
    if (animal->UNPREG==1){
        Sterile++;
    }
} else if ( ((LDBS2 + PTEST)>365) && (currentDATE==((LDBS2 + PTEST) - 365)) &&
    ( (animal->SEAOP1 + animal->SEAOP2)==2) ){
    animal->FAIL++;
    animal->SEAOP1=0;
    animal->SEAOP2=0;
    CowsFail++;
    if (animal->UNPREG==1){
        Sterile++;
    }
}

else if ( (TEST==1) && (animal->SEX==3) && (animal->Puberty == 1) && (animal->INELIG==0)){
    if (NumBreedSeasns==1){
        if (SpringCalfSeasn==1){
            if ( (currentDATE==(LDBS + PTEST))){
                if ((animal->PREG==0) && (animal->INELIG == 0)){
                    animal->FAIL++;
                    CowsFail++;
                    if (animal->UNPREG==1){
                        Sterile++;
                    }
                }
            }
        }
    }
}
else if (FallCalfSeasn==1) {
    if ( ((LDBS2 + PTEST)<=365) && (currentDATE==(LDBS2 + PTEST))) {
        if ((animal->PREG==0) && (animal->INELIG == 0)) {
            animal->FAIL++;
            CowsFail++;
            if (animal->UNPREG==1) {
                Sterile++;
            }
        }
    }
    else if ( ((LDBS2 + PTEST)>365) && (currentDATE==((LDBS2 + PTEST) - 365)) &&
              (simulationDAY>(((LDBS2 + PTEST)-365)) && (LDBS2<=365)) {
        if ((animal->PREG==0) && (animal->INELIG == 0)) {
            animal->FAIL++;
            CowsFail++;
            if (animal->UNPREG==1) {
                Sterile++;
            }
        }
    }
    else if ( ((LDBS2 + PTEST)>365) && (currentDATE==((LDBS2 + PTEST) - 365)) &&
              (simulationDAY>(((LDBS2 + PTEST)-365)) && (LDBS2>365)) {
        if ((animal->PREG==0) && (animal->INELIG == 0)) {
            animal->FAIL++;
            CowsFail++;
            if (animal->UNPREG==1) {
                Sterile++;
            }
        }
    }
    else if (NumBreedSeasns==2) {
        if (SpringCalfSeasn==1) {
            if ( (currentDATE==(LDBS + PTEST)) ) {
                if ((animal->PREG==0) && (animal->INELIG == 0)) {
                    animal->SEAOP1=1;
                }
            }
        }
        else if (FallCalfSeasn==1) {
            if ( ( (LDBS2 + PTEST) <=365) && (currentDATE==(LDBS2 + PTEST)) ) {
                if ((animal->PREG==0) && (animal->INELIG == 0)) {
                    animal->SEAOP2=1;
                }
            }
        }
        else if ( ( (LDBS2 + PTEST) >365) && (currentDATE==(LDBS2 + PTEST)) ) {
            if ((animal->PREG==0) && (animal->INELIG == 0)) {
                animal->SEAOP2=1;
            }
        }
    }
}
PTEST - 365))
&& (simulationDAY>((LDBS2 + PTEST)-365)) && (LDBS2<=365))
if (((animal->PREG==0) && (animal->INELIG == 0)){
animal->SEAOP2=1;
}
}
else if ( ((LDBS2 + PTEST) >365) && (currentDATE==(LDBS2+
PTEST - 365))
&& (simulationDAY>((LDBS2 + PTEST)-365)) && (LDBS2>365)){
if ((animal->PREG==0) && (animal->INELIG == 0)){
animal->SEAOP2=1;
}
}
if ( ((LDBS2 + PTEST)<=365) && (currentDATE==365) &&  ((animal->SEAOP1 + animal->SEAOP2)==2) ){
animal->FAIL++;
animal->SEAOP1=0;
animal->SEAOP2=0;
CowsFail++;
if (animal->UNPREG==1){
Sterile++;
}
}
else if ( ((LDBS2 + PTEST)>365) && (currentDATE==((LDBS2+
PTEST) - 365)) &&  ((animal->SEAOP1 + animal->SEAOP2)==2) ){
animal->FAIL++;
animal->SEAOP1=0;
animal->SEAOP2=0;
CowsFail++;
if (animal->UNPREG==1){
Sterile++;
}
}
void Cow::Puberty(AnimalRecord *animal){
if ( (animal->CALFSEAS==1) && (animal->SEX == 0) && (animal->
REPLA == 1) )
{ if ( ( (animal->BREED < 7) | | (animal->BREED==8) | | (animal-
>BREED==9) | |(animal->BREED == 12) ) && (animal->CDWT >= .6 * (double)animal->PMWT) )
{ animal->SEX=3;
animal->WEANWT=animal->CDWT;
if (animal->WEAN==0){
animal->WEAN=animal->AGE;
}
TSWEANAGE += animal->WEAN;
SCALF++;
}
if (animal->WEANDATE == 0)
{
    animal->WEANDATE = currentDate;
}
TNCOH--;
animal->PADG = animal->PADG - animal->MHETADG;
animal->REPLA = 0;
animal->PUBER = currentDate;
animal->Puberty = 1;
} else if (
    (((animal->BREED >= 10) && (animal->BREED < 12))
    || (animal->BREED == 7))
    &&
    (animal->CDWT >= .55 * (double)animal->PMWT)
)
{
animal->SEX = 3;
animal->WEANWT = animal->CDWT;
if (animal->WEAN == 0)
{
    animal->WEAN = animal->AGE;
    TSWEANAGE += animal->WEAN;
    SCALF++;
}
if (animal->WEANDATE == 0)
{
    animal->WEANDATE = currentDate;
}
TNCOH--;
animal->PADG = animal->PADG - animal->MHETADG;
animal->REPLA = 0;
animal->PUBER = currentDate;
animal->Puberty = 1;
}
else if (
    (animal->CALTSEAS == 2) && (animal->SEX == 0) &&
    (animal->REPLA == 1)
)
{
    if ((
        (animal->BREED < 5) || (animal->BREED == 12)
    ) &&
    (animal->CDWT >= .6 * (double)animal->PMWT)
    )
    {
        animal->SEX = 3;
animal->WEANWT = animal->CDWT;
        if (animal->WEAN == 0)
        {
            animal->WEAN = animal->AGE;
            TFWEANAGE += animal->WEAN;
            FCAFLF++;
        }
        if (animal->WEANDATE == 0)
        {
            animal->WEANDATE = currentDate;
        }
        TNCOH--;
animal->PADG = animal->PADG - animal->MHETADG;
}
animal->REPLA=0;
animal->PUBER = currentDATE;
animal->Puberty=1;
}
else if ( ((animal->BREED >= 5) && (animal->BREED < 12)) &&
  (animal->CDWT >= .55 * (double)animal->PMWT) )
{
  animal->SEX=3;
animal->WEANWT=animal->CDWT;
if (animal->WEAN==0){
animal->WEAN=animal->AGE;
TFWEANAGE=animal->WEAN;
FCALF++;
}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
TNCOH--;
animal->PADG=animal->PADG - animal->MHETADG;
animal->REPLA=0;
animal->PUBER = currentDATE;
animal->Puberty=1;
}
else if ( (animal->SEX==4) && (animal->REPLA == 1) )
{
  if ( ( (animal->BREED < 5) || (animal->BREED == 12) ) &&
  (animal->CDWT >= .6 * (double)animal->PMWT) )
  {
    TNCOH--;
animal->SEX=3;
animal->REPLA=0;
animal->PUBER = currentDATE;
animal->Puberty=1;
  }
else if ( ((animal->BREED >= 5) && (animal->BREED < 12)) &&
  (animal->CDWT >= .55 * (double)animal->PMWT) )
  {
    TNCOH--;
animal->SEX=3;
animal->REPLA=0;
animal->PUBER = currentDATE;
animal->Puberty=1;
  }
else if ( (animal->SEX == 3) && (animal->REPLA == 1) )
{
  if ( ( (animal->BREED < 5) || (animal->BREED == 12) ) &&
  (animal->CDWT >= .6 * (double)animal->PMWT) )
  {
    animal->SEX=3;
animal->REPLA=0;
animal->PUBER = currentDATE;
  }
animal->Puberty=1;
}
else if ( ((animal->BREED >= 5) && (animal->BREED < 12) ) && (animal->CDWT >= 0.55 * (double)animal->PMWT) )
{
    animal->SEX=3;
    animal->REPLA=0;
    animal->PUBER = currentDate;
    animal->Puberty=1;
}
}

void Cow::Dystocia(AnimalRecord *animal){
    double randomnumber = rand();
    double MCDYSF = (randomnumber/(double) RAND_MAX);
    double ACWM2 = 0;
    double ACWF2 = 0;
    double ACWM3 = 0;
    double ACWF3 = 0;
    double ACWM4 = 0;
    double ACWF4 = 0;
    double DYPerM2 = 0.43485;
    double DYPerF2 = 0.28403;
    double DYPerM3 = 0.13719;
    double DYPerF3 = 0.05917;
    double DYPerM4 = 0.04568;
    double DYPerF4 = 0.02618;
    double DYSTF=0;
    double DYPerM21 = 0.2178;
    double DYPerF21 = 0.14224;
    double DYPerM31 = 0.0687;
    double DYPerF31 = 0.0296;
    double DYPerM41 = 0.0229;
    double DYPerF41 = 0.0131;
    if ((animal->CSEX==2) && (animal->AGE < 913)){
        ACWM2 = (animal->ACBWT) * 0.937035033 * 1.045398773;
    }
    else if ((animal->CSEX==0) && (animal->AGE < 913)){
        ACWF2 = (animal->ACBWT) * 0.937035033 * 0.954601227;
    }
    else if ((animal->CSEX==2) && ((animal->AGE >= 913) && (animal->AGE < 1278))){
        ACWM3 = (animal->ACBWT) * 1.017854727 * 1.039601329;
    }
    else if ((animal->CSEX==0) && ((animal->AGE >= 913) && (animal->AGE < 1278))){
        ACWF3 = (animal->ACBWT) * 1.017854727 * 0.960398671;
    }
    else if ((animal->CSEX==2) && (animal->AGE >= 1278)){
        ACWM4 = (animal->ACBWT) * 1.045177871 * 1.050084121;
} else if ((animal->CSEX==0) && (animal->AGE >= 1278)){
  ACWF4 = (animal->ACBWT) * 1.045177871 * .949915879;
}

} else if ( (animal->PREG==1) && (animal->DPREG >= animal->PGEST) && (CALFEASE==0)) {
  if ( (animal->AGE < 913) && (animal->CSEX == 2) ){
    DYSTF = DYPerM2 + (0.01 * ( (animal->CBWT) - ACWM2));
  }

  else if ( (animal->AGE < 913) && (animal->CSEX == 0) ){
    DYSTF = DYPerF2 + (0.01 * ( (animal->CBWT) - ACWF2));
  }

  else if ( ((animal->AGE >= 913) && (animal->AGE < 1278)) && (animal->CSEX == 2) ){
    DYSTF = DYPerM3 + (0.01 * ( (animal->CBWT) - ACWM3));
  }

  else if ( ((animal->AGE >= 913) && (animal->AGE < 1278)) && (animal->CSEX == 0) ){
    DYSTF = DYPerF3 + (0.01 * ( (animal->CBWT) - ACWF3));
  }

  else if ( (animal->AGE >= 1278) && (animal->CSEX == 2) ){
    DYSTF = DYPerM4 + (0.01 * ( (animal->CBWT) - ACWM4));
  }

  else if ( (animal->AGE >= 1278) && (animal->CSEX == 0) ){
    DYSTF = DYPerF4 + (0.01 * ( (animal->CBWT) - ACWF4));
  }

} else if ( (animal->PREG==1) && (animal->DPREG >= animal->PGEST) && (CALFEASE==1)) {
  if ( (animal->AGE < 913) && (animal->CSEX == 2) ){
    DYSTF = DYPerM21 + (0.01 * ( (animal->CBWT) - ACWM2));
  }

  else if ( (animal->AGE < 913) && (animal->CSEX == 0) ){
    DYSTF = DYPerF21 + (0.01 * ( (animal->CBWT) - ACWF2));
  }

  else if ( ((animal->AGE >= 913) && (animal->AGE < 1278)) && (animal->CSEX == 2) ){
    DYSTF = DYPerM31 + (0.01 * ( (animal->CBWT) - ACWM3));
  }

  else if ( ((animal->AGE >= 913) && (animal->AGE < 1278)) && (animal->CSEX == 0) ){
    DYSTF = DYPerF31 + (0.01 * ( (animal->CBWT) - ACWF3));
  }

  else if ( (animal->AGE >= 1278) && (animal->CSEX == 2) ){
    DYSTF = DYPerM41 + (0.01 * ( (animal->CBWT) - ACWM4));
  }

  else if ( (animal->AGE >= 1278) && (animal->CSEX == 0) ){
    DYSTF = DYPerF41 + (0.01 * ( (animal->CBWT) - ACWF4));
  }

if (MCDYSF <= DYSTF) {
    animal->DYSTO = 1;
} else {
    animal->DYSTO = 0;
}
if ((animal->AGE<1095) && (animal->DYSTO==1)) {
    Dystocia2++;
} else if ((animal->AGE>=1095) && (animal->AGE<1460) && (animal->DYSTO==1)) {
    Dystocia3++;
} else if ((animal->AGE>=1460) && (animal->DYSTO==1)) {
    Dystocia4++;
}

void Cow::BreedFemales(AnimalRecord *animal) {
    double MCCYC = GetStochasticNumber(1.0, 21);
    double MCPREG = (rand()/(double) RAND_MAX);
    double randomnumber=rand();
    double MCDET = (randomnumber/(double) RAND_MAX);
    static double NODET = 0.05;
    double CONCV = .75;
    double FESTR=0;
    AnimalRecord *tempcow = new AnimalRecord;
    if ((SpringCalfSeasn==1) && (currentDATE == FDBHS-1)) {
        NumHeifers=0;
    } else if ((SpringCalfSeasn==1) && (currentDATE == FDBS-1)) {
        NumCows=0;
    } else if ((FallCalfSeasn==1) && (currentDATE == FDBHS2-1)) {
        NumHeifers=0;
    } else if ((FallCalfSeasn==1) && (currentDATE == FDBS2-1)) {
        NumCows=0;
    } if (SpringCalfSeasn==1) {
        if ((currentDATE==FDBHS) && (animal->PARIE==0) && (animal->AGE>=56/7) && (animal->SEX==3)) {
            NumHeifers++;
        } else if ((currentDATE==FDBS) && (animal->PARIE>0) && (animal->SEX==3)) {
            NumCows++;
        }
if (FallCalfSeasn==1) {
if ((currentDATE==FDBHS2) & (animal->PARIE==0) & (animal->AGE>=56/7) & (animal->SEX==3)) {
    NumHeifers++;
} else if ((currentDATE==(FDBHS2 +(2*(SEAS2/3)) - 365)) & (animal->PARIE==0) & (animal->AGE>=56/7) & (((animal->SEX==3) || ((animal->SEX==0) & (animal->REPLA==1)) || ((animal->SEX==4) & (animal->REPLA==1))) & ((FDBHS2 +(2*(SEAS2/3))) >365 ) ){
    NumHeifers++;
} else if ((currentDATE==FDBS2) & (animal->PARIE>0) & (animal->SEX==3)) {
    NumCows++;
} }

if ((SpringCalfSeasn==1) & (currentDATE == FDBHS)){
    Servi=0;
    CONCE=0;
    CowsInElig = 0;
    CowsElig=0;
    CyclHeifer1=0;
    CyclHeifer2=0;
    CyclHeifer3=0;
    PregHef1=0;
    PregHef2=0;
    PregHef3=0;
    CyclCow1=0;
    CyclCow2=0;
    CyclCow3=0;
    PregCow1=0;
    PregCow2=0;
    PregCow3=0;
} else if ((FallCalfSeasn==1) & (currentDATE ==FDBHS2-1)){
    Servi1=Servi;
    CONCE1=CONCE;
    CowsInElig1=CowsInElig;
    CowsElig1=CowsElig;
    CyclHeifer21=0;
    CyclHeifer22=0;
    CyclHeifer23=0;
    PregHef21=0;
    PregHef22=0;
    PregHef23=0;
    CyclCow21=0;
    CyclCow22=0;
    CyclCow23=0;
    PregCow21=0;
    PregCow22=0;
PregCow23=0;
}
else if ((FallCalfSeasn==1) && (currentDATE == FDBHS2))
{
  Servi=0;
  CONCE=0;
  CowsInElig = 0;
  CowsElig=0;
}
else if ((FallCalfSeasn==1) && (currentDATE == FDBHS2))
{
  Servi=0;
  CONCE=0;
  CowsInElig = 0;
  CowsElig=0;
}

if ( (animal->PARIE == 0) && ( (animal->Puberty == 1) &&
  ((animal->AGE/7)>=56) ) )
{
  animal->INELIG=0;
}

if ( (SpringCalfSeasn == 1) && (currentDATE == LDHBS) &&
  (animal->PARIE == 0) &&
  (animal->SEX==3) ||
  ((animal->SEX==3) && (animal->SEX==0) && (animal->REPLA==1)) ||
  ((animal->SEX==4) && (animal->REPLA==1)) )
{
  if (animal->INELIG==1)
  {
    CowsInElig++;
  }
  else if (animal->INELIG ==0)
  {
    CowsInElig++;
  }
}
else if ((FallCalfSeasn == 1) && (LDHBS2<=365) && (currentDATE == LDHBS2) &&
  (animal->PARIE == 0) &&
  (animal->SEX==3) ||
  ((animal->SEX==3) && (animal->SEX==0) && (animal->REPLA==1)) ||
  ((animal->SEX==4) && (animal->REPLA==1)) )
{
  if (animal->INELIG==1)
  {
    CowsInElig++;
  }
  else if (animal->INELIG ==0)
  {
    CowsInElig++;
  }
}
else if ((FallCalfSeasn == 1) && (LDHBS2>365) && (currentDATE == LDHBS2) &&
  (animal->PARIE == 0) &&
  (animal->SEX==3) ||
  ((animal->SEX==3) && (animal->SEX==0) && (animal->REPLA==1)) ||
  ((animal->SEX==4) && (animal->REPLA==1)) )
{
  if (animal->INELIG==1)
  {
    CowsInElig++;
  }
  else if (animal->INELIG ==0)
  {
    CowsInElig++;
  }
}
else if ((SpringCalfSeasn == 1) && (currentDATE == LDBS) &&
  (animal->PARIE > 0) &&
  (animal->SEX==3) )
{
CowsElig++;  
}  
else if ((FallCalfSeasn == 1) && (LDBS2<=365) && (currentDATE == LDBS2) && (animal->PARIE > 0) && (animal->SEX==3) )  
{  
CowsElig++;  
}  
else if ((FallCalfSeasn == 1) && (LDBS2>365) && (currentDATE== LDBS2-365) && (animal->PARIE > 0) && (animal->SEX==3) )  
{  
CowsElig++;  
}  

if ((animal->BCS<=4)){  
CONCV=.75 * .6824 * (1 + (0.115 + (0.01 * (int (animal->AGE)/365))) * .6111);  
}  
else if ((animal->BCS==5)){  
CONCV=.75 * 1.0 * (1 + (0.115 + (0.01 * (int (animal->AGE)/365))) * .6111);  
}  
else if ((animal->BCS>=6)){  
CONCV=.75 * 1.1177 * (1 + (0.115 + (0.01 * (int (animal->AGE)/365))) * .6111);  
}  

if ((SpringCalfSeasn==1) && (YEAR>=0)){  
if ((currentDATE==(FDBHS +(SEAS/3))) && (animal->PARIE==0) && ((animal->AGE/7)>=56) && (animal->UNPREG==0)){  
CyclHeifer1++;  
if (animal->PREG==1){  
PregHef1++;  
}  
}  
else if ((currentDATE==(FDBHS +(2*(SEAS/3)))) && (animal->PARIE==0) && (animal->AGE/7>=56) && (animal->UNPREG==0)){  
CyclHeifer2++;  
if (animal->PREG==1){  
PregHef2++;  
}  
}  
else if ((currentDATE==(FDBS +(SEAS/3))) && (animal->PARIE>0) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0)){  
CyclCow1++;  
if (animal->PREG==1){  
PregCow1++;  
}  
}  
else if ((currentDATE==(FDBS +(SEAS/3))) && (animal->PARIE>0) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0)){  
CyclCow1++;  
if (animal->PREG==1){  
PregCow1++;  
}  
}  

460
PregCOW1++;
}
}
else if ((currentDATE==(FDBS +(2*(SEAS/3)))) && (animal->PARIE>0) && (currentDATE>animal->CADAT+animal->PPI) &&
(animal->UNPREG==0) ){  
CyclCow2++;  
if (animal->PREG==1){  
PregCow2++;
}
else if ((currentDATE==(FDBS +SEAS)) && (animal->PARIE>0) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0) ){  
CyclCow3++;  
if (animal->PREG==1){  
PregCow3++;
}
if ((FallCalfSeasn==1) && (FDBHS2+SEAS2<=365) && (FDBS2+SEAS2<=365) && (YEAR>=0)){  
if ((currentDATE==(FDBHS2 +(SEAS2/3))) && (animal->PARIE==0) && (animal->AGE/7>=56) && (animal->UNPREG==0)){  
CyclHeifer21++;  
if (animal->PREG==1){  
PregHef21++;
}
else if ((currentDATE==(FDBHS2 +(2*(SEAS2/3)))) && (animal->PARIE==0) && (animal->AGE/7>=56) && (animal->UNPREG==0)){  
CyclHeifer22++;  
if (animal->PREG==1){  
PregHef22++;
}
else if ((currentDATE==(FDBS2 +SEAS2)) && (animal->PARIE==0) && (animal->AGE/7>=56) && (animal->UNPREG==0)){  
CyclHeifer23++;  
if (animal->PREG==1){  
PregHef23++;
}
else if ((currentDATE==(FDBS2 +(SEAS2/3))) && (animal->PARIE>0) && (currentDATE>animal->CADAT+animal->PPI) &&
(animal->UNPREG==0) ){  
CyclCow21++;  
if (animal->PREG==1){  
PregCow21++;
}
}
else if ((currentDATE==(FDBS2 + (2*(SEAS2/3)))) && (animal->PARIE>0) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0) ){
  CyclCow22++;
  if (animal->PREG==1){
    PregCow22++;
  }
}
else if ((currentDATE==(FDBS2+SEAS2)) && (animal->PARIE>0) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0) ){
  CyclCow23++;
  if (animal->PREG==1){
    PregCow23++;
  }
}
else if ((FallCalfSeasn==1) && (FDBHS2+SEAS2<=365) && (FDBS2+SEAS2>365) && (YEAR>=0)){
  if ((currentDATE==(FDBHS2 + (SEAS2/3))) && (animal->PARIE==0) && (animal->AGE/7>=56) && (animal->UNPREG==0)){
    CyclHeifer21++;
    if (animal->PREG==1){
      PregHef21++;
    }
  }
  else if ((currentDATE==(FDBHS2 + (2*(SEAS2/3)))) && (animal->PARIE==0) && (animal->AGE/7>=56) && (animal->UNPREG==0)){
    CyclHeifer22++;
    if (animal->PREG==1){
      PregHef22++;
    }
  }
  else if ((currentDATE==(FDBS2 +SEAS2)) && (animal->PARIE==0) && (animal->AGE/7>=56) && (animal->UNPREG==0)){
    CyclHeifer23++;
    if (animal->PREG==1){
      PregHef23++;
    }
  }
  else if ((currentDATE==(FDBS2 +(SEAS2/3))) && (animal->PARIE>0) && ((FDBS2 +(SEAS2/3))<=365) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0) ){
    CyclCow21++;
    if (animal->PREG==1){
      PregCow21++;
    }
  }
  else if ((currentDATE==(FDBS2 +(SEAS2/3)- 365)) && (animal->PARIE>0) && ((FDBS2 +(SEAS2/3))>365) && (currentDATE>animal->CADAT+animal->PPI-365) && (animal->UNPREG==0) )

CyclCow21++;  
if (animal->PREG==1){
PregCow21++;  
}
}

else if ((currentDATE==(FDBS2 + (2*(SEAS2/3)))) && (animal->PARIE>0) && ((FDBS2 + (2*(SEAS2/3)))<=365) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0)){
CyclCow22++;  
if (animal->PREG==1){
PregCow22++;  
}
}

else if ((currentDATE==((FDBS2 +(2*(SEAS2/3)) - 365))) && (animal->PARIE>0) && ((FDBS2 +(2*(SEAS2/3)))>365) && (currentDATE>animal->CADAT+animal->PPI-365) && (animal->UNPREG==0)){
CyclCow23++;  
if (animal->PREG==1){
PregCow23++;  
}
}

else if ((FallCalfSeasn==1) && (FDBHS2+SEAS2>365) && (FDBS2+SEAS2>365) && (YEAR>=0)){
if ((currentDATE==(FDBHS2 +(SEAS2/3))) && (animal->PARIE==0) && (animal->AGE/7>=56) && ((FDBHS2 + (SEAS2/3))<=365) && (animal->UNPREG==0)){
CyclHeifer21++;  
if (animal->PREG==1){
PregHef21++;  
}
}

else if ((currentDATE==(FDBHS2 +(SEAS2/3)) - 365)) && (animal->PARIE==0) && (animal->AGE/7>=56) && ((FDBHS2 + (SEAS2/3))>365) && (animal->UNPREG==0)){
CyclHeifer21++;  
if (animal->PREG==1){
PregHef21++;  
}
}

else if ((currentDATE==((FDBHS2 + (2*(SEAS2/3))) - 365)) && (animal->PARIE==0) && (animal->AGE/7>=56) && ((FDBHS2 + (2*(SEAS2/3)))<=365) && (animal->UNPREG==0)){
CyclHeifer21++;  
if (animal->PREG==1){
PregHef21++;  
}
}

else if ((currentDATE==((FDBHS2 + (2*(SEAS2/3))) - 365)) && (animal->PARIE==0) && (animal->AGE/7>=56) && ((FDBHS2 + (2*(SEAS2/3)))>365) && (animal->UNPREG==0)){
CyclHeifer21++;  
if (animal->PREG==1){
PregHef21++;  
}
}
PARIE==0) && (animal->AGE/7>=56) && ((FDBHS2+(2*(SEAS2/3)))<=365) && (animal->UNPREG==0))
CyclHeifer22++;
if (animal->PREG==1){
PregHef22++;
}
else if ((currentDATE==(FDBHS2+(2*(SEAS2/3)) - 365)) &&
(animal->PARIE==0) && (animal->AGE/7>=56) && ((FDBHS2+(2*(SEAS2/3)))>365) && (animal->UNPREG==0))
CyclHeifer22++;
if (animal->PREG==1){
PregHef22++;
}
else if ((currentDATE==(FDBHS2 +SEAS2 - 365)) && (animal->PARIE==0) && (animal->AGE/7>=56) && ((FDBHS2 +SEAS2)>365) && (animal->UNPREG==0))
CyclHeifer23++;
if (animal->PREG==1){
PregHef23++;
}
if ((currentDATE==(FDBS2 +(SEAS2/3))) && (animal->PARIE>0) && ((FDBS2 +(SEAS2/3))<=365) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0) )
CyclCow21++;
if (animal->PREG==1){
PregCow21++;
}
else if ((currentDATE==(FDBS2 +(SEAS2/3)) - 365)) && (animal->PARIE>0) && ((FDBS2 +(SEAS2/3))>365) && (currentDATE>animal->CADAT+animal->PPI-365) && (animal->UNPREG==0) )
CyclCow21++;
if (animal->PREG==1){
PregCow21++;
}
else if ((currentDATE==(FDBS2 +(2*(SEAS2/3)))) && (animal->PARIE>0) && ((FDBS2 +(2*(SEAS2/3)))<=365) && (currentDATE>animal->CADAT+animal->PPI) && (animal->UNPREG==0) )
CyclCow22++;
if (animal->PREG==1){
PregCow22++;
}
else if ((currentDATE==(FDBS2 +(2*(SEAS2/3)) - 365)) && (animal->PARIE>0) && ((FDBS2 +(2*(SEAS2/3)))>365) && (currentDATE>animal->CADAT+animal->PPI-365) && (animal->UNPREG==0) )
CyclCow22++;
if (animal->PREG==1){
PregCow22++;
}
if (animal->PREG==1)
    PregCow22++;
}

else if (((currentDATE==(FDBS2+ SEAS2 - 365)) && (animal->PARIE>0) && ((FDBS2+SEAS2)>365) && (currentDATE>animal->CADAT+animal->PPI-365) && (animal->UNPREG==0))
    CyclCow23++;
else if (animal->PREG==1)
    PregCow23++;
}

if ( (currentDATE == 1) && (animal->Puberty==1) ){
    if ( ((animal->PARIE>0) || (animal->PARIE==0) ) && (animal->CADAT==0) && (animal->CUCYC>=365) && (animal->PUBER==1) )
        animal->CUCYC=((animal->CUCYC) - 365);
    animal->CNDAT=0;
    animal->PUBER = 1;
}
else if ( (animal->PARIE>0) && ( animal->CADAT>0) && ((animal->CADAT + animal->CUCYC)>=365) )
    animal->CUCYC= ((animal->CUCYC + animal->CADAT) - 365);
else if ( (animal->PARIE==0) && ( animal->PUBER>1) &&
    ((animal->PUBER + animal->CUCYC)>=365) )
    animal->CUCYC= ((animal->CUCYC) - 365);
else if ( (animal->PARIE==0) && ( animal->PUBER==1) &&
    (animal->CUCYC>=365) )
    animal->CUCYC= ((animal->CUCYC) - 365);
else{
    animal->CNDAT=0;
    animal->CUCYC=0;
    animal->PUBER = 1;
}

if ( (currentDATE == 1) && ( ((animal->CADAT) + (animal->PPI)) <= 365 ) )
    animal->CADAT=0;
    animal->PPI=0;
else if ( (currentDATE == 1) && ( (animal->CADAT) + (animal->PPI)) >= 365 ) ){
    animal->PPI= ( (animal->CADAT) + (animal->PPI)) - 365 );
    animal->CADAT=0;
}
if ( (animal->SEX == 3) && (animal->PREG == 0) && (animal->UNPREG == 0) )
    (animal->PUBER==1) && ( currentDATE >= ((animal->CADAT) +
    (animal->CUCYC) + 1)) ){
    animal->CUCYC += MCCYC;
} else if ( (animal->SEX == 3) && (animal->PREG == 0) &&
    (animal->UNPREG == 0) &&
    (animal->PUBER>1) && ( currentDATE >= ((animal->PUBER) +
    (animal->CUCYC) + 1 )) )
    animal->CUCYC += MCCYC;
} if ( (animal->SEX==3) && (SpringCalfSeasn==1) && (currentDATE==FDBHS) && (animal->UNPREG==0) )
    animal->SERVI=0;
    animal->SEAOP1=0;
else if ( (animal->SEX==3) && (FallCalfSeasn==1) && (currentDATE==FDBHS2) && (animal->UNPREG==0) )
    animal->SERVI=0;
    animal->SEAOP2=0;
} else if ( (animal->SEX == 3) && (animal->PREG == 0) && (animal->UNPREG == 0) && (animal->CULL==0) )
    if ( (animal->PARIE == 0)  && (animal->Puberty==1) &&
        (current DATE >=animal->PUBER) )
        if  (animal->PUBER>1)
            FESTR = animal->PUBER + animal->CUCYC;
        else if ( (animal->Puberty==1) && (animal->PUBER==1) )
            FESTR=animal->CUCYC;
        if  (NumBreedSeasns==1){
            if ( (SpringCalfSeasn==1) && (currentDATE==FDBHS) && (currentDATE<=LDHBS) )
                if((FESTR >= FDBHS) && (FESTR <= LDHBS) && (current-
                    DATE==(int) FESTR + 1) )
                if (MCDET >= NODET)
                    animal->DETEC = 1;
            else{
                animal->DETEC = 0;
            }
        } else if (animal->DETEC == 1){
            animal->SERVI++; 
            Servi++;
if (MCPREG <= (CONCV) ){
    animal->PREG = 1;
    animal->CNDAT = FESTR;
    animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
    PNUM++;
    CONCE++;
    tempcow->DAMBREED = animal->BREED;
    BirthAnimal(tempcow, animal);
}
else{
    animal->PREG = 0;
}
}
}
}

else if ( (FallCalfSeasn==1) && (currentDATE>=FDBHS2) &&
    (currentDATE<=LDHBS2) && (LDHBS2<=365) && (FDBHS2<=LDHBS2) ){
    if( (FESTR >= FDBHS2) && (FESTR <= LDHBS2) &&
        (currentDATE==(int)FESTR + 1) ){
        if (MCDET >= NODET){
            animal->DETEC = 1;
        }
        else{
            animal->DETEC = 0;
        }
    }
    else{
        animal->DETEC = 0;
    }
    if (animal->DETEC == 1){
        animal->SERVI++;
        Servi++;
        if (MCPREG <= (CONCV) ){
            animal->PREG = 1;
            animal->CNDAT = FESTR;
            animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
            PNUM++;
            CONCE++;
            tempcow->DAMBREED = animal->BREED;
            BirthAnimal(tempcow, animal);
        }
        else{
            animal->PREG = 0;
        }
    }
}
else if ( (FallCalfSeasn==1) && (FDBHS2<=LDHBS2) &&
    (currentDATE>=FDBHS2) && (currentDATE<=365) && (LDHBS2>365) ){
    if( (FESTR >= FDBHS2) && (FESTR <= 365) &&
        (currentDATE==(int)FESTR + 1) ){
        if (MCDET >= NODET){
            animal->DETEC = 1;
        }
        else{
            animal->DETEC = 0;
        }
        if (animal->DETEC == 1){
            animal->SERVI++;
            Servi++;
            if (MCPREG <= (CONCV) ){
                animal->PREG = 1;
                animal->CNDAT = FESTR;
                animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
                PNUM++;
                CONCE++;
                tempcow->DAMBREED = animal->BREED;
                BirthAnimal(tempcow, animal);
            }
            else{
                animal->PREG = 0;
            }
        }
    }
}
if (MCDET >= NODET){
animal->DETEC = 1;
}
else{
animal->DETEC = 0;
}
if (animal->DETEC == 1){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV) ){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal-
>PGAVG));
PNUM++; 
CONCE++;
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else{
animal->PREG = 0;
}
}
else if ( (FallCalfSeasn==1) && (FDBHS2<=LDHBS2) &&
(currentDATE<=(LDHBS2-365)) && (currentDATE>=1) &&
(LDHBS2>365) ){
if( (FESTR >= 1) && (FESTR <= (LDHBS2-365)) && (current-
DATE==(int)FESTR + 1) ){
if (MCDET >= NODET){
animal->DETEC = 1;
}
else{
animal->DETEC = 0;
}
if (animal->DETEC == 1){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV) ){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal-
>PGAVG));
PNUM++; 
CONCE++;
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else{
animal->PREG = 0;
}
else if (!(NumBreedSeasns==2) && ((currentDATE>=FDBHS) && currentDATE<=LDHBS) || ((currentDATE>=FDBHS2) && currentDATE<=LDHBS2) && (FDBHS2< LDHBS2) && (LDHBS2<=365) }){
    if ((FESTR >= FDBHS) && (FESTR <= LDHBS) && (currentDATE==(int)FESTR + 1) ){
        if (MCDET >= NODET){
            animal->DETEC = 1;
        } else{
            animal->DETEC = 0;
        }
    if (animal->DETEC == 1){
        animal->SERVI++; Servi++;
        if (MCPREG <= (CONCV) ){
            animal->PREG = 1;
            animal->CNAT = FESTR;
            animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
            PNUM++;
            CONCE++;
            tempcow->DAMBREED = animal->BREED;
            BirthAnimal(tempcow, animal);
        } else{
            animal->PREG = 0;
        }
    }
    else if ((FESTR >= FDBHS2) && (FESTR <= LDHBS2) && (LDHBS2<=365) && (currentDATE==(int)FESTR + 1) ){
        if (MCDET >= NODET){
            animal->DETEC = 1;
        } else{
            animal->DETEC = 0;
        }
    if (animal->DETEC == 1){
        animal->SERVI++; Servi++;
        if (MCPREG <= (CONCV) ){
            animal->PREG = 1;
            animal->CNAT = FESTR;
            animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
            PNUM++;
            CONCE++;
            tempcow->DAMBREED = animal->BREED;
            BirthAnimal(tempcow, animal);
        } else{
            animal->PREG = 0;
        }
    } else if (FESTR >= FDBHS2) && (FESTR <= LDHBS2) && (LDHBS2<=365) && (currentDATE==(int)FESTR + 1) ){
        if (MCDET >= NODET){
            animal->DETEC = 1;
        } else{
            animal->DETEC = 0;
        }
    if (animal->DETEC == 1){
        animal->SERVI++; Servi++;
        if (MCPREG <= (CONCV) ){
            animal->PREG = 1;
            animal->CNAT = FESTR;
            animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
            PNUM++;
            CONCE++;
            tempcow->DAMBREED = animal->BREED;
            BirthAnimal(tempcow, animal);
        } else{
            animal->PREG = 0;
        }
    } else if (FESTR >= FDBHS2) && (FESTR <= LDHBS2) && (LDHBS2<=365) && (currentDATE==(int)FESTR + 1) ){
        if (MCDET >= NODET){
            animal->DETEC = 1;
        } else{
            animal->DETEC = 0;
        }
    if (animal->DETEC == 1){
        animal->SERVI++; Servi++;
        if (MCPREG <= (CONCV) ){
            animal->PREG = 1;
            animal->CNAT = FESTR;
            animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
            PNUM++;
            CONCE++;
            tempcow->DAMBREED = animal->BREED;
            BirthAnimal(tempcow, animal);
        } else{
            animal->PREG = 0;
        }
    }
PNUM++;  
CONCE++;  
tempcow->DAMBREED = animal->BREED;  
BirthAnimal(tempcow, animal);  
}  
else{  
animal->PREG = 0;  
}  
}  
}  
}  
else if ( (NumBreedSeasns==2) && ( (currentDATE>=FDBHS) 
&
(currentDATE<=LDHBS) ) || ( (currentDATE==FDBHS2) &&
(currentDATE<=365) && (FDBHS2<=LDHBS2) && (LDHBS2>=365)) ||
( (currentDATE>=1) && (currentDATE<=LDHBS2 - 365 )
&& (LDHBS2>=365)) )
if ( (FESTR >= FDBHS) && (FESTR <= LDHBS) && (current-
DATE==(int)FESTR + 1) )
if (MCDET >= NODET)
animal->DETEC = 1;  
else{  
animal->DETEC = 0;  
}  
if (animal->DETEC == 1){  
animal->SERVI++;  
Servi++;  
if (MCPREG <= (CONCV) )
animal->PREG = 1;  
animal->CNDAT = FESTR;  
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->
>PGAVG));  
PNUM++;  
CONCE++;  
tempcow->DAMBREED = animal->BREED;  
BirthAnimal(tempcow, animal);  
}  
else{  
animal->PREG = 0;  
}  
}  
}  
else if ( (FESTR >= FDBHS2) && (FESTR <= 365) &&
(FDBHS2<=LDHBS2) &&(LDHBS2>=365) && (currentDATE==(int)FESTR
+ 1) )
if (MCDET >= NODET)
animal->DETEC = 1;  
else{  
animal->DETEC = 0;
if (animal->DETEC == 1){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV) ){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
PNUM++;
CONCE++;
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else{
animal->PREG = 0;
}
}
else if ( (FESTR >= 1) && (FESTR <= (LDHBS2-365)) &&
(LDHBS2>=365) &&
(FDBHS2 <=LDHBS2) &&(currentDATE==(int)FESTR + 1) ){ if (MCDET >= NODET){
animal->DETEC = 1;
}
else{
animal->DETEC = 0;
}
if (animal->DETEC == 1){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV) ){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
PNUM++;
CONCE++;
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else{
animal->PREG = 0;
}
}
}
else if ( (animal->PARIE > 0) && (currentDATE >= animal->CADAT + animal->PPI ) ){ FESTR = ( (animal->CADAT) + (animal->CUCYC) ) ;
if (NumBreedSeasns==1){
if ( (SpringCalfSeasn==1) && (currentDATE>=FDBS) && (currentDATE<=LDBS) ){
if ( (FESTR >= FDBS) && (FESTR <= LDBS) && (currentDATE == (int)FESTR + 1) ){
if (MCDET >= NODET){
animal->DETEC = 1;
}
else{
animal->DETEC = 0;
}
if ( (animal->DETEC == 1) && (animal->DYSTO == 0) ){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV)){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
PNUM++;
if (PNUM>CONCE){
CONCE++;
}
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else{
animal->PREG = 0;
}
}else if ( (animal->DETEC == 1) && (animal->DYSTO == 1) ){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV - 0.061)){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
PNUM++;
if (PNUM>CONCE){
CONCE++;
}
}
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else if  ( (FallCalfSeasn==1) && (currentDATE>=FDBS2) && (currentDATE<=LDBS2) && (LDBS2<=365) ){
if ( (FESTR >= FDBS2) && (FESTR <= LDBS2) && (LDBS2<=365) && 
(currentDATE==(int)FESTR + 1) ){ 
if (MCDET >= NODET){
animal->DETEC = 1;
}
else{
animal->DETEC = 0;
}
if ( (animal->DETEC == 1) && (animal->DYSTO == 0) ){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV) ){ 
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
PNUM++;
if (PNUM>CONCE){
CONCE++;
}
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else{
animal->PREG = 0;
}
}
else if ( (animal->DETEC == 1) && (animal->DYSTO == 1) ){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV- 0.061)){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
PNUM++;
if (PNUM>CONCE){
CONCE++;
}
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
}
else if ( (FallCalfSeasn==1) && (LDBS2>365) && 
(current-DATE=FDBS2) && (currentDATE<=365) ){
if((FESTR >= FDBS2) && (FESTR <= 365) && (current- 
DATE==(int)FESTR + 1)) { 
if (MCDET >= NODET){
animal->DETEC = 1;
if ( (animal->DETEC == 1) && (animal->DYSTO == 0) ){
    animal->SERVI++;
    Servi++;
    if (MCPREG <= (CONCV) ){
        animal->PREG = 1;
        animal->CNDAT = FESTR;
        animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
        PNUM++;
        if (PNUM>CONCE){
            CONCE++;
        }
        tempcow->DAMBREED = animal->BREED;
        BirthAnimal(tempcow, animal);
    }
    else{
        animal->PREG = 0;
    }
}
else if ( (animal->DETEC == 1) && (animal->DYSTO == 1) ){
    animal->SERVI++;
    Servi++;
    if (MCPREG <= (CONCV- 0.061)){
        animal->PREG = 1;
        animal->CNDAT = FESTR;
        animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
        PNUM++;
        if (PNUM>CONCE){
            CONCE++;
        }
        tempcow->DAMBREED = animal->BREED;
        BirthAnimal(tempcow, animal);
    }
    else{
        animal->PREG = 0;
    }
}
else if ( (FallCalfSeasn==1) && (LDBS2>365) && (current-DATE>=1 ) && (currentDATE<=(LDBS2-365)) ){
    if( (FESTR >= 1) && (FESTR <= (LDBS2 -365)) ) && (current-DATE==((int)FESTR + 1) ){
        if (MCDET >= NODET){
            animal->DETEC = 1;
        }
    else{
        animal->DETEC = 0;
    }
if ( (animal->DETEC == 1) && (animal->DYSTO == 0) ){
    animal->SERVI++;
    Servi++;
    if (MCPREG <= (CONCV)) {
        animal->PREG = 1;
        animal->CNDAT = FESTR;
        animal->PGEST = GetStochasticNumber((animal->PGSD), (animal->PGAVG));
        PNUM++;
        if (PNUM >= CONCE) {
            CONCE++;
        }
        tempcow->DAMBREED = animal->BREED;
        BirthAnimal(tempcow, animal);
    } else {
        animal->PREG = 0;
    }
} else if ( (animal->DETEC == 1) && (animal->DYSTO == 1) ) {
    animal->SERVI++;
    Servi++;
    if (MCPREG <= (CONCV - 0.061)) {
        animal->PREG = 1;
        animal->CNDAT = FESTR;
        animal->PGEST = GetStochasticNumber((animal->PGSD), (animal->PGAVG));
        PNUM++;
        if (PNUM >= CONCE) {
            CONCE++;
        }
        tempcow->DAMBREED = animal->BREED;
        BirthAnimal(tempcow, animal);
    } else {
        animal->PREG = 0;
    }
} else if ( (NumBreedSeasns == 2) && ( (currentDATE >= FDBS) &&
    (currentDATE <= LDBS) || (currentDATE >= FDBS2) &&
    (currentDATE <= LDBS2) && (LDBS2 <= 365) ) ) {
    if ( (FESTR >= FDBS) && (FESTR <= LDBS) &&
    (currentDATE == (int)FESTR + 1) ) {
        if (MCDET >= NODET){
            animal->DETEC = 1;
        } else {
            animal->DETEC = 0;
        }
    } if ( (animal->DETEC == 1) && (animal->DYSTO == 0) ) {
        animal->SERVI++;
    }
Servi++;  
if (MCPREG <= (CONCV) ){  
    animal->PREG = 1;  
    animal->CNDAT = FESTR;  
    animal->PGEST = GetStochasticNumber((animal->PGSD) , (animal->PGAVG));  
    PNUM++;  
    if (PNUM>CONCE){  
        CONCE++;  
    }  
    tempcow->DAMBREED = animal->BREED;  
    BirthAnimal(tempcow, animal);  
}  
else{  
    animal->PREG = 0;  
}  
}  
else if ( (animal->DETEC == 1) & (animal->DYSTO == 1) ){  
    animal->SERVI++;  
    Servi++;  
    if (MCPREG <= (CONCV - 0.061)){  
        animal->PREG = 1;  
        animal->CNDAT = FESTR;  
        animal->PGEST = GetStochasticNumber((animal->PGSD) , (animal->PGAVG));  
        PNUM++;  
        if (PNUM>CONCE){  
            CONCE++;  
        }  
        tempcow->DAMBREED = animal->BREED;  
        BirthAnimal(tempcow, animal);  
    }  
}  
else if ( (FESTR >= FDBS2) & (FESTR <= LDBS2) & (current.DATE== (int)FESTR + 1) ){  
    if (MCDET >= NODET){  
        animal->DETEC = 1;  
    }  
    else{  
        animal->DETEC = 0;  
    }  
    if ( (animal->DETEC == 1) & (animal->DYSTO == 0) ){  
        animal->SERVI++;  
        Servi++;  
        if (MCPREG <= (CONCV) ){  
            animal->PREG = 1;  
            animal->CNDAT = FESTR;  
            animal->PGEST = GetStochasticNumber((animal->PGSD) , (animal->PGAVG));  
            PNUM++;  
        }  
    }  
}
if (PNUM>CONCE) {
    CONCE++;
} else {
    animal->PREG = 0;
}
else if ( animal->DETEC == 1) && (animal->DYSTO == 1) ) {
    animal->SERVI++;
    Servi++;
    if (MCPREG <= (CONCV - 0.061)) {
        animal->PREG = 1;
        animal->CNDAT = FESTR;
        animal->PGEST = GetStochasticNumber((animal->PGSD), (animal->PGAVG));
        PNUM++;
        if (PNUM>CONCE) {
            CONCE++;
        }
    }
    tempcow->DAMBREED = animal->BREED;
    BirthAnimal(tempcow, animal);
}
else if ( NumBreedSeasns==2) && ( (currentDATE>=FDBS) && (currentDATE<=LDBS) ) || ( (currentDATE>=FDBS2) && (currentDATE<=365) && (LDBS2>=365) ) || ( (currentDATE>1) && (currentDATE<=LDBS2) && (LDBS2>=365) ) ) {
    if ( (FESTR >= FDBS) && (FESTR <= LDBS) && (currentDATE==(int)FESTR + 1) ) {
        if (MCDET >= NODET) {
            animal->DETEC = 1;
        } else {
            animal->DETEC = 0;
        }
        if ( animal->DETEC == 1) && (animal->DYSTO == 0) ) {
            animal->SERVI++;
            Servi++;
            if (MCPREG <= (CONCV) ) {
                animal->PREG = 1;
                animal->CNDAT = FESTR;
                animal->PGEST = GetStochasticNumber((animal->PGSD), (animal->PGAVG));
                PNUM++;
            }
            if (PNUM>CONCE) {
                CONCE++;
            }
        }
    } else {
        animal->DETEC = 1;
        if ( animal->DETEC == 1) && (animal->DYSTO == 0) ) {
            animal->SERVI++;
            Servi++;
            if (MCPREG <= (CONCV) ) {
                animal->PREG = 1;
                animal->CNDAT = FESTR;
                animal->PGEST = GetStochasticNumber((animal->PGSD), (animal->PGAVG));
                PNUM++;
            }
            if (PNUM>CONCE) {
                CONCE++;
            }
        }
    }
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else{
animal->PREG = 0;
}
}
else if ( (animal->DETEC == 1) && (animal->DYSTO == 1) ){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV - 0.061)){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
PNUM++;
if (PNUM>CONCE){
CONCE++;
}
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
}
else if ( (FESTR >= FDBS2) && (FESTR <= 365) && (LDBS2>=365)
&& (currentDATE==(int)FESTR + 1) ){  
if (MCDET >= NODET){
animal->DETEC = 1;
}
else{
animal->DETEC = 0;
}
if ( (animal->DETEC == 1) && (animal->DYSTO == 0) ){
animal->SERVI++;
Servi++;
if (MCPREG <= (CONCV) ){
animal->PREG = 1;
animal->CNDAT = FESTR;
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->PGAVG));
PNUM++;
if (PNUM>CONCE){
CONCE++;
}
tempcow->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
else{
animal->PREG = 0;
}
else if ( (animal->DETEC == 1) && (animal->DYSTO == 1) ){
    animal->SERVI++;
    Servi++;
    if (MCPREG <= (CONCV - 0.061)){
        animal->PREG = 1;
        animal->CNDAT = FESTR;
        animal->PGEST = GetStochasticNumber((animal->PGSD), (animal->PGAVG));
        PNUM++;
        if (PNUM>CONCE){
            CONCE++;
        }
        tempcow->DAMBREED = animal->BREED;
        BirthAnimal(tempcow, animal);
    }
    else if ( (FESTR >= 1) && (FESTR <= (LDBS2-365)) &&
            (LDBS2>=365) && (currentDATE==(int)FESTR + 1) ){
        if (MCDET >= NODET){
            animal->DETEC = 1;
        }
        else{
            animal->DETEC = 0;
        }
        if ( (animal->DETEC == 1) && (animal->DYSTO == 0) ){
            animal->SERVI++;
            Servi++;
            if (MCPREG <= (CONCV)){
                animal->PREG = 1;
                animal->CNDAT = FESTR;
                animal->PGEST = GetStochasticNumber((animal->PGSD), (animal->PGAVG));
                PNUM++;
                if (PNUM>CONCE){
                    CONCE++;
                }
                tempcow->DAMBREED = animal->BREED;
                BirthAnimal(tempcow, animal);
            }
            else{
                animal->PREG = 0;
            }
        }
        else if ( (animal->DETEC == 1) && (animal->DYSTO == 1) ){
            animal->SERVI++;
            Servi++;
            if (MCPREG <= (CONCV - 0.061)){
                animal->PREG = 1;
                animal->CNDAT = FESTR;
            }
animal->PGEST=GetStochasticNumber((animal->PGSD) , (animal->
>PGAVG));
PNUM++;
if (PNUM>CONCE){
CONCE++;
}
tempre->DAMBREED = animal->BREED;
BirthAnimal(tempcow, animal);
}
}
}
}
}
}
delete tempcow;

void Cow::Parturition(AnimalRecord *animal){
double MCPPI = GetStochasticNumber(12.5, 54.5);
AnimalRecord *tempcow;
if ( (animal->PREG == 1) && (animal->PGEST <= animal->DPREG) ){
NCTB++;
animal->PARIIE++;
tempre = GetAnimal(animal->FIDN);
animal->CIDN=(animal->FIDN);
animal->CBIRTHDATE=currentDATE;
animal->CUCYC = 0;
while (MCPPI<30){
if (MCPPI< 30){
MCPPI = GetStochasticNumber(12.5, 54.4);
}
MCPPI=MCPPI;
}
animal->PPI = MCPPI;
if (animal->AGE<=(3 * 365)){
animal->PPI= animal->PPI + 7;
}
if (animal->DYSTO==1){
animal->PPI=animal->PPI + 14;
}
animal->LACT = 1;
animal->CADAT = currentDATE;
TCDAT = TCDAT + animal->CADAT;
CNUMB++;
TotGestLen+=animal->PGEST;
animal->PREG = 0;
animal->DPREG = 0;
PNUM--;
animal->LDAY = 1;
animal->CDWTK = ((animal->SBWTK - (animal->GUWTK))/ 0.96);
animal->CDWT = animal->CDWTK * 2.20459;
animal->SBWTK= animal->CDWTK * .96;
animal->EBWTK = animal->SBWTK * .891;
animal->EQSBW=animal->SBWTK * (478/((animal->PMWT*.453599) * .96));
animal->EQEBW=animal->EQSBW * .891;
animal->GUWTK = 0;
animal->FIDN = 0;
animal->CBWT=0;
animal->CSEX=0;
TNCOH++;
if (tempcow != 0){
tempcow->AGE=1;
TBWT+=tempcow->BWT;
animal->PGEST = 0;
if (tempcow->SEX==0){
TBWTH+=tempcow->BWT;
NumAvailRepla++;
CNUMBH++;
}
else if (tempcow->SEX==2){
TBWTS+=tempcow->BWT;
CNUMBS++;
}
tempcow->BIRTHDATE=currentDATE;
if ( (NumBreedSeasns==1) && (SpringCalfSeasn==1) ){
tempcow->CALFSEAS=1;
}
else if ( (NumBreedSeasns==1) && (FallCalfSeasn==1) ){
tempcow->CALFSEAS=2;
}
else if ( (NumBreedSeasns==2) && (currentDATE<182) ){
tempcow->CALFSEAS=1;
}
else if ( (NumBreedSeasns==2) && (currentDATE>=182) ){
tempcow->CALFSEAS=2;
}
else if (tempcow ==0){
NCTB--;
animal->PARIE--;
animal->CIDN=0;
animal->CBIRTHDATE=0;
animal->CUCYC = 0;
animal->LACT = 0;
animal->PPI =0;
TCDAT = TCDAT - animal->CADAT;
CNUMB--;
TotGestLen-=animal->PGEST;
animal->PGEST = 0;
animal->LDAY = 0;
void Cow::Mortality (AnimalRecord *animal){
    double randomnumber=rand();
    double MCDIE = (randomnumber/(double) RAND_MAX);
    AnimalRecord *tempcow;
    if ( (animal->CALTSEAS==1) && ((WEANSDATE - (animal->AGE + animal->BIRTHDATE)) >= 0 ) && (animal->SEX<3) && (EZBREED !=1) && (EZCALF != 1) ){
        if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<= WEANSDATE) && (animal->REPLA==0) ){
            if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
            }
            else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
            }
            else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
            }
            else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
            }
            else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
            }
        }
        else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<= WEANSDATE) && (animal->REPLA==0) ){
            if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
            }
            else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
            }
            else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
            }
            else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
            }
            else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
            }
    }
}
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE 
<= 0.00105525) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && 
(MCDIE <= 0.0002166) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && 
(MCDIE <= 0.00006365) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) && (MCDIE <= 0.000034144) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;  
DNUMH++;  
NumAvailRepla--;  
TNCOH--;  
}  
else{  
animal->DIE = 0;  
}  
if (animal->DIE == 1){  
tempcow = GetAnimal(animal->MIDN);  
tempcow->LACT=0;  
tempcow->LDAY=0;  
tempcow->MCMFC=0;  
tempcow->MCSNF=0;  
tempcow->YN=0;  
tempcow->YEN=0;  
tempcow->CIDN=0;  
tempcow->CBIRTHDATE=0;  
tempcow->FWEAN++;  
}  
}  
else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) && (animal->REPLA==1) ){  
if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}  
else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}  
else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}  
else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}  
484
DNUM++; 
DNUMH++; 
TNCOH--; 
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--; 
} 
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--; 
} 
else if ( (animal->AGE > 6) && (animal->AGE <= 9) &&
    (MCDIE <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--; 
} 
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--; 
} 
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
    (MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--; 
} 
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
    (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
DNUM++;  
DNUMH++;  
TNCOH--;  
}  
else if ( (animal->AGE > 101) && ((animal->AGE +  
animal->BIRTHDATE) <= WEANSDATE) && (MCDIE <= 0.000034144)  ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else{
    animal->DIE = 0;  
}
if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);  
tempcow->LACT = 0;  
tempcow->LDAY = 0;  
tempcow->MCMFC=0;  
tempcow->MCSNF=0;  
tempcow->YN=0;  
tempcow->YEN=0;  
tempcow->CIDN=0;  
tempcow->CBIRTHDATE=0;  
tempcow->FWEAN++;  
CowsFailWean++;  
}
else if ( (animal->SEX == 2) && ((animal->AGE +  
animal->BIRTHDATE) <= WEANSDATE) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
        animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
        animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
        animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
        animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
        animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) )  
{  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) )  
{  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) )  
{  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE  
<= 0.00105525) )  
{  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) )  
{  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&  
(MCDIE <= 0.0002166) )  
{  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&  
(MCDIE <= 0.00006365) )  
{  
animal->DIE = 1;  
animal->AGED = animal->AGE;
DNUM++; 
DNUMS++; 
TNCOH--; }
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) && (MCDIE <= 0.00034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++; 
    DNUMS++; 
    TNCOH--; }
else{
    animal->DIE = 0; }
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
}

else if ( (animal->CALFSEAS==1) && ((WEANSDATE - (animal->AGE + animal->BIRTHDATE)) >= 0 ) && (animal->SEX<3) && (EZBREED ==1) && (EZCALF != 1) ){
    if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) && (animal->REPLA==0) ){
        if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++; 
            DNUMH++; 
            NumAvailRepla--; 
            TNCOH--; }
    else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++; 
        DNUMH++; 
        NumAvailRepla--; 
        TNCOH--; ;
else if ((animal->AGE == 3) && (MCDIE <= 0.00179)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}

else if ((animal->AGE == 4) && (MCDIE <= 0.00183)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}

else if ((animal->AGE == 5) && (MCDIE <= 0.00134)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}

else if ((animal->AGE == 6) && (MCDIE <= 0.000804)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}

else if ((animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}

else if ((animal->AGE == 10) && (MCDIE <= 0.000737)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}

else if ((animal->AGE > 10) && (animal->AGE <= 41) &&
(MCDIE <= 0.0002166) }
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
(MCDIE <= 0.00006365) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
}else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= WEANSDATE) && (MCDIE <= 0.000034144) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
}else{
animal->DIE = 0;
}
if (animal->DIE == 1) {
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
}
else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<= WEANSDATE) && (animal->REPLA==1) ){
if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;

}
else if ((animal->AGE == 2) && (MCDIE <= 0.00322)){
   animal->DIE = 1;
   animal->AGED = animal->AGE;
   REPLR++;
   DNUM++;
   DNUMH++;
   TNCOH--;
}
else if ((animal->AGE == 3) && (MCDIE <= 0.00179)){
   animal->DIE = 1;
   animal->AGED = animal->AGE;
   REPLR++;
   DNUM++;
   DNUMH++;
   TNCOH--;
}
else if ((animal->AGE == 4) && (MCDIE <= 0.00183)){
   animal->DIE = 1;
   animal->AGED = animal->AGE;
   REPLR++;
   DNUM++;
   DNUMH++;
   TNCOH--;
}
else if ((animal->AGE == 5) && (MCDIE <= 0.00134)){
   animal->DIE = 1;
   animal->AGED = animal->AGE;
   REPLR++;
   DNUM++;
   DNUMH++;
   TNCOH--;
}
else if ((animal->AGE == 6) && (MCDIE <= 0.000804)){
   animal->DIE = 1;
   animal->AGED = animal->AGE;
   REPLR++;
   DNUM++;
   DNUMH++;
   TNCOH--;
}
else if ((animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525)){
   animal->DIE = 1;
   animal->AGED = animal->AGE;
   REPLR++;
   DNUM++;
   DNUMH++;
   TNCOH--;
}
else if ((animal->AGE == 10) && (MCDIE <= 0.000737)){


animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
(MCDIE <= 0.0002166) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
(MCDIE <= 0.00006365) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= WEANSDATE) && (MCDIE <= 0.000034144) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else{
animal->DIE = 0;
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
else if ( (animal->SEX == 2) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        NUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE < 0.00105525) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
            (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else{
    animal->DIE = 0;
}
}
else if ( (animal->CALFSEAS==1) && ((WEANSDATE -


(animal->AGE + animal->BIRTHDATE) >= 0 ) && (animal->SEX<3) && ((EZBREED ==1) || (EZBREED != 1)) && (EZCALF == 1) ){
    if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<= WEANSDATE) && (animal->REPLA==0) ){
        if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
    }
}
}    else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}    else{
    animal->DIE = 0;
}
    if (animal->DIE == 1){
        tempcow = GetAnimal(animal->MIDN);
        tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
}

else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<= WEANSDATE)  && (animal->REPLA==1) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){  
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
    }  
else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){  
      animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
    }  
else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){  
      animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
    }  
else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){  
      animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
    }  
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){  
      animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
    else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
    }
    else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
    }
    else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= WEANSDATE) && (MCDIE <= 0.000034144) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
        DNUM++;
}
DNUMH++;  
TNCOH--;  
}
else{
    animal->DIE = 0;
}
if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
    tempcow->MCMFC = 0;
    tempcow->MCSNF = 0;
    tempcow->YN = 0;
    tempcow->YEN = 0;
    tempcow->CIDN = 0;
    tempcow->CBIRTHDATE = 0;
    tempcow->FWEAN++;
    CowsFailWean++;
}
    else if ( (animal->SEX == 2) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) ){  
        if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMS++;
            TNCOH--; 
        }
    }else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--; 
    }
else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--; 
    }
else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--; 
    }
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){

animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--; 
} else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--; 
} else if ( (animal->AGE > 6) && (animal->AGE <= 9) &&
(MCDIE <= 0.00105525) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--; 
} else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--; 
} else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
(MCDIE <= 0.0002166) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--; 
} else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
(MCDIE <= 0.00006365) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--; 
} else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= WEANSDATE) && (MCDIE <= 0.000034144) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--; 
}


```c

else{
    animal->DIE = 0;
}

if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
    tempcow->MCMFC = 0;
    tempcow->MCSNF = 0;
    tempcow->YN = 0;
    tempcow->YEN = 0;
    tempcow->CIDN = 0;
    tempcow->CBIRTHDATE = 0;
    tempcow->FWEAN++;
    CowsFailWean++;
}
else if ((animal->CalfSea == 1) && ((WeansDate - (animal->AGE + animal->BIRTHDATE)) < 0) && (animal->SEX < 3) && (EZBreed != 1) && (EZCalf != 1)){
    if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= (WeansDate + 365)) && (animal->REPLA == 0)) {
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMH++;
        NumAvailRepla--;
        TNCOH--;
    }
    else if ( (animal->AGE == 2) && (MCDIE <= 0.03618) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMH++;
        NumAvailRepla--;
        TNCOH--;
    }
    else if ( (animal->AGE == 3) && (MCDIE <= 0.019438) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMH++;
        NumAvailRepla--;
        TNCOH--;
    }
    else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
```
DNUM++; 
DNUMH++; 
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++; 
    DNUMH++; 
    NumAvailRepla--; 
    TNCOH--; 
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++; 
    DNUMH++; 
    NumAvailRepla--; 
    TNCOH--; 
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE
           <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++; 
    DNUMH++; 
    NumAvailRepla--; 
    TNCOH--; 
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++; 
    DNUMH++; 
    NumAvailRepla--; 
    TNCOH--; 
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
           (MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++; 
    DNUMH++; 
    NumAvailRepla--; 
    TNCOH--; 
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
           (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= (WEANSDATE + 365)) && (MCDIE <= 0.000034144) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
} else{
animal->DIE = 0;
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= (WEANSDATE + 365)) && (animal->REPLA==1) ){
if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
} else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
} else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) &&
(MCDIE <= 0.00105525) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
(MCDIE <= 0.0002166) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
(MCDIE <= 0.00006365) ){ 
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE > 101) && ((animal->AGE +
animal->BIRTHDATE) <= (WEANSDATE + 365)) && (MCDIE <=
0.000034144) ){ 
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else{
animal->DIE = 0;
}
if (animal->DIE == 1){
  tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
}
else if (animal->SEX == 2) && ((animal->AGE +
animal->BIRTHDATE) <= (WEANSDATE + 365)) { 
  if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){ 
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--;
}
else if ((animal->AGE == 2) && (MCDIE <= 0.003618)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ((animal->AGE == 3) && (MCDIE <= 0.0019438)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ((animal->AGE == 4) && (MCDIE <= 0.00201)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ((animal->AGE == 5) && (MCDIE <= 0.00134)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ((animal->AGE == 6) && (MCDIE <= 0.000804)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ((animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ((animal->AGE == 10) && (MCDIE <= 0.000737)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ((animal->AGE > 10) && (animal->AGE <= 41) &&
(MCDIE <= 0.0002166) }
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  DNUM++;
  DNUMS++;
  TNCOH--;
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && 
  (MCDIE <= 0.00006365) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  DNUM++;
  DNUMS++;
  TNCOH--;
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->
  BIRTHDATE) <= (WEANSDATE + 365)) && (MCDIE <= 0.000034144) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  DNUM++;
  DNUMS++;
  TNCOH--;
}
else{
  animal->DIE = 0;
}
if (animal->DIE == 1){
  tempcow = GetAnimal(animal->MIDN);
  tempcow->LACT = 0;
  tempcow->LDAY = 0;
  tempcow->MCMFC=0;
  tempcow->MCSNF=0;
  tempcow->YN=0;
  tempcow->YEN=0;
  tempcow->CIDN=0;
  tempcow->CBirthDATE=0;
  tempcow->FWEAN++;
  CowsFailWean++;
}
else if ((animal->CALFSEAS==1) && ((WEANSDATE - (an-
 imal->AGE + animal->BIRTHDATE)) < 0 ) && (animal->SEX<3) &&
  (EZBREED ==1) && (EZCALF != 1)){
  if ( (animal->SEX == 0) && ((animal->AGE + animal->
  BIRTHDATE) <= (WEANSDATE + 365)) && (animal->REPLA==0) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
      animal->DIE = 1;
      animal->AGED = animal->AGE;
      DNUM++;
    }
DNUMH++;  
NumAvailRepla--;  
TNCOH--;  
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;  
    TNCOH--;  
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;  
    TNCOH--;  
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;  
    TNCOH--;  
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;  
    TNCOH--;  
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;  
    TNCOH--;  
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;  
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
            (MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
            (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= (WEANSDATE + 365)) &&
            (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;}
else{
    animal->DIE = 0;
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= (WEANSDATE + 365)) && (animal->REPLA==1) ){
  if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }
  else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }
  else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }
  else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }
  else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }
  else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
  }
DNUMH++; 
TNCOH--; 
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++; 
  DNUM++; 
  DNUMH++; 
  TNCOH--; 
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++; 
  DNUM++; 
  DNUMH++; 
  TNCOH--; 
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++; 
  DNUM++; 
  DNUMH++; 
  TNCOH--; 
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++; 
  DNUM++; 
  DNUMH++; 
  TNCOH--; 
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= (WEANSDATE + 365)) && (MCDIE <= 0.000034144) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++; 
  DNUM++; 
  DNUMH++; 
  TNCOH--; 
}
else{
  animal->DIE = 0;
}
if (animal->DIE == 1) {
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
    tempcow->MCMFC = 0;
    tempcow->MCSNF = 0;
    tempcow->YN = 0;
    tempcow->YEN = 0;
    tempcow->CIDN = 0;
    tempcow->CBIRTHDATE = 0;
    tempcow->FWEAN++;
    CowsFailWean++;
}
}

    else if ( (animal->SEX == 2) && ((animal->AGE + animal->BIRTHDATE) <= (WEANSDATE + 365)) ){
        if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMS++;
            TNCOH--;
        }
        else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMS++;
            TNCOH--;
        }
        else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMS++;
            TNCOH--;
        }
        else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMS++;
            TNCOH--;
        }
        else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMS++;
            TNCOH--;
        }
    }
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= (WEANSDATE + 365)) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}
else{
    animal->DIE = 0;
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
}
else if ((animal->CalfSEAS==1) && ((WEANSDATE - (animal->AGE + animal->BIRTHDATE)) < 0 ) && (animal->SEX<3) && ((EZBREED == 1) || (EZBREED !=1)) && (EZCALF == 1)){
    if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<= (WEANSDATE + 365)) && (animal->REPLA==0 )){
        if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){  
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){ 
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
        else if ( (animal->AGE == 5) && (MCDIE <= 0.00159) ){ 
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
    }
    else { 
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMH++;
        NumAvailRepla--;
        TNCOH--;
    }
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
}
else if ( (animal->AGE > 101) && ((animal->AGE +
animal->BIRTHDATE)<= (WEANSDATE + 365)) && (MCDIE <=
0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;}
else{
    animal->DIE = 0;
}
if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT=0;
    tempcow->LDAY=0;
    tempcow->MCMFC=0;
    tempcow->MCSNF=0;
    tempcow->YN=0;
    tempcow->YEN=0;
    tempcow->CIDN=0;
    tempcow->CBIRTHDATE=0;
    tempcow->FWEAN++;
    CowsFailWean++;
}
else if ( (animal->SEX == 0) && ((animal->AGE + animal->
BIRTHDATE)<= (WEANSDATE + 365)) && (animal->REPLA==1) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;}
    else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;}
    else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
TNCOH--; 
} else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--; 
} else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--; 
} else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--; 
} else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE
  <= 0.00105525) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--; 
} else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--; 
} else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE
  <= 0.0002166) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--; 
  else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
  return; 
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= (WEANSDATE + 365)) && (MCDIE <= 0.00034144) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
} else {
    animal->DIE = 0;
}
if (animal->DIE == 1) {
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
    tempcow->MCMFC = 0;
    tempcow->MCSNF = 0;
    tempcow->YN = 0;
    tempcow->YEN = 0;
    tempcow->CIDN = 0;
    tempcow->CBIRTHDATE = 0;
    tempcow->FWEAN++;
    CowsFailWean++;
}
else if ( (animal->SEX == 2) && ((animal->AGE + animal->BIRTHDATE) <= (WEANSDATE + 365)) ) {
    if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ) {
        animal->DIE = 1;
       动物->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    } else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ) {
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    } else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ) {
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        } else {   // More else if conditions here
TNCOH--;  
}  
else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ) {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ) {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ) {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ) {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ) {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ) {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ) {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
          (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}  
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= (WEANSDATE + 365)) &&
          (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
}  
else{  
animal->DIE = 0;
}  
if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
    tempcow->MCMFC=0;
    tempcow->MCSNF=0;
    tempcow->YN=0;
    tempcow->YEN=0;
    tempcow->CIDN=0;
    tempcow->CBIRTHDATE=0;
    tempcow->FWEAN++;
    CowsFailWean++;
}
}

else if ( (animal->CALFSEAS==2) && ((WEANFDATE- (animal->AGE + animal->BIRTHDATE)) >= 0 ) &&
          (animal->SEX<3) &&
          (EZBREED !=1) && (EZCALF != 1)){
    if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<= WEANFDATE) &&
       (animal->REPLA==0) ){
        if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
            animal->DIE = 1;
            animal->AGED = animal->AGE;
            DNUM++;
            DNUMH++;
            NumAvailRepla--;
            TNCOH--;
        }
    }
else if ((animal->AGE == 2) && (MCDIE <= 0.003618)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ((animal->AGE == 3) && (MCDIE <= 0.0019438)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ((animal->AGE == 4) && (MCDIE <= 0.00201)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ((animal->AGE == 5) && (MCDIE <= 0.00134)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ((animal->AGE == 6) && (MCDIE <= 0.000804)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
    else if ((animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525)) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ((animal->AGE == 10) && (MCDIE <= 0.000737)) {
    animal->DIE = 1;
}
animal->AGED = animal->AGE;
DNUM++;  
DNUMH++;  
NumAvailRepla--;  
TNCOH--;  
}  
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&  
(MCDIE <= 0.0002166) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMH++;  
NumAvailRepla--;  
TNCOH--;  
}  
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&  
(MCDIE <= 0.00006365) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMH++;  
NumAvailRepla--;  
TNCOH--;  
}  
else if ( (animal->AGE > 101) && ((animal->AGE +  
animal->BIRTHDATE)<= WEANFDATE) && (MCDIE <= 0.000034144) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMH++;  
NumAvailRepla--;  
TNCOH--;  
}  
else{  
animal->DIE = 0;  
}  
if (animal->DIE == 1){  
tempcow = GetAnimal(animal->MIDN);  
tempcow->LACT=0;  
tempcow->LDAY=0;  
tempcow->MCMFC=0;  
tempcow->MCSNF=0;  
tempcow->YN=0;  
tempcow->YEN=0;  
tempcow->CIDN=0;  
tempcow->CBIRTHDATE=0;  
tempcow->FWEAN++;  
CowsFailWean++;  
}  
}  
}
else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= WEANFDATE) && (animal->REPLA==1) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
    }
    else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
    }
    else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
    }
    else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
    }
    else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
    }
    else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
        DNUMH++;
        TNCOH--;
    }
    else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE
<= 0.00105525) }
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }else if ( (animal->AGE > 10) && (animal->AGE <= 41) && MCDIE
               <= 0.0002166) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
               (MCDIE <= 0.0006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }else if ( (animal->AGE > 101) && ((animal->AGE +
    animal->BIRTHDATE) <= WEANFDATE) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
  }else{
    animal->DIE = 0;
  }
  if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}

    else if ( (animal->SEX == 2) && ((animal->AGE +
animal->BIRTHDATE)<= WEANFDATE) ){
        if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
            animal->DIE = 1;
animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
        }
        else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
            animal->DIE = 1;
animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
        }
        else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
            animal->DIE = 1;
animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
        }
        else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
            animal->DIE = 1;
animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
        }
        else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
            animal->DIE = 1;
animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
        }
        else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
            animal->DIE = 1;
animal->AGED = animal->AGE;
        DNUM++;
        }

525
DNUMS++;  
TNCOH--;  

} else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;  
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;  
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;  
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;  
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= WEANFDATE) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;  
}
else{
    animal->DIE = 0;
}

if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
    tempcow->MCMFC=0;
    tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
}
}

else if ((animal->CALFSEAS==2) && ((WEANFDATE - (animal->AGE + animal->BIRTHDATE)) < 0 ) && (animal->SEX<3) && (EZBREED !=1) && (EZCALF != 1)){
    if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<=(WEANFDATE + 365)) && (animal->REPLA==0) ){animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;

527
NumAvailRepla--;  
TNCOH--;  
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
         (MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
         (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;

DNUM++;  
DNUMH++;  
NumAvailRepla--;  
TNCOH--;  
}
else  
{
  animal->DIE = 0;  
}
if (animal->DIE == 1){  
tempcow = GetAnimal(animal->MIDN);  
tempcow->LACT=0;  
tempcow->LDAY=0;  
tempcow->MCMFC=0;  
tempcow->MCSNF=0;  
tempcow->YN=0;  
tempcow->YEN=0;  
tempcow->CIDN=0;  
tempcow->CBIRTHDATE=0;  
tempcow->FWEAN++;  
CowsFailWean++;  
}
else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (animal->REPLA==1) ){
  if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
    animal->DIE = 1;  
aminal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
  animal->DIE = 1;  
aminal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
  animal->DIE = 1;  
aminal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
  animal->DIE = 1;  
aminal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
  animal->DIE = 1;  
aminal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  

530
REPLR++; 
DNUM++; 
DNUMH++; 
TNCOH--; 
} 
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (MCDIE <= 0.000034144)) {
    animal->DIE = 1; 
    animal->AGED = animal->AGE;
    REPLR++; 
    DNUM++; 
    DNUMH++; 
    TNCOH--; 
} 
else{
    animal->DIE = 0;
} 
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN); 
tempcow->LACT = 0; 
tempcow->LDAY = 0; 
tempcow->MCMFC=0; 
tempcow->MCSNF=0; 
tempcow->YN=0; 
tempcow->YEN=0; 
tempcow->CIDN=0; 
tempcow->CBIRTHDATE=0; 
tempcow->FWEAN++; 
CowsFailWean++; 
} 
else if ( (animal->SEX == 2) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.0385) ){
        animal->DIE = 1; 
        animal->AGED = animal->AGE; 
        DNUM++; 
        DNUMS++; 
        TNCOH--; 
    } 
    else if ( (animal->AGE == 2) && (MCDIE <= 0.003618) ){
        animal->DIE = 1; 
        animal->AGED = animal->AGE;
        DNUM++; 
        DNUMS++; 
        TNCOH--; 
    } 
    else if ( (animal->AGE == 3) && (MCDIE <= 0.0019438) ){
        animal->DIE = 1; 
        animal->AGED = animal->AGE;
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00201) ){
    animal->DIE = 1;  
    animal->AGED = animal->AGE;  
    DNUM++;  
    DNUMS++;  
    TNCOH--;  
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;  
    animal->AGED = animal->AGE;  
    DNUM++;  
    DNUMS++;  
    TNCOH--;  
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;  
    animal->AGED = animal->AGE;  
    DNUM++;  
    DNUMS++;  
    TNCOH--;  
    else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
        animal->DIE = 1;  
        animal->AGED = animal->AGE;  
        DNUM++;  
        DNUMS++;  
        TNCOH--;  
    }
    else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
        animal->DIE = 1;  
        animal->AGED = animal->AGE;  
        DNUM++;  
        DNUMS++;  
        TNCOH--;  
    }
    else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
        animal->DIE = 1;  
        animal->AGED = animal->AGE;  
        DNUM++;  
        DNUMS++;  
        TNCOH--;  
    }
    else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
        animal->DIE = 1;  
        }
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--;
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (MCDIE <= 0.000034144) ) {
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--;
}
else{
animal->DIE = 0;
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC = 0;
tempcow->MCSNF = 0;
tempcow->YN = 0;
tempcow->YEN = 0;
tempcow->CIDN = 0;
tempcow->CBIRTHDATE = 0;
tempcow->FWEAN++;
CowsFailWean++;
}
}
else if ( (animal->CalfSEAS == 2) && ((WEANFDATE- (animal->AGE + animal->BIRTHDATE)) >= 0 ) && (animal->SEX < 3) && (EZBREED == 1) && (EZCALF != 1)){
  if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= WEANFDATE) && (animal->REPLA == 0) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
      animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--;
    }
    else if ( (animal->AGE == 2) && (MCDIE <= 0.0322) ){
      animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
    }
TNCOH--; 
} 
else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--; 
    TNCOH--; 
} 
else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--; 
    TNCOH--; 
} 
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--; 
    TNCOH--; 
} 
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--; 
    TNCOH--; 
} 
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE 
    <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--; 
    TNCOH--; 
} 
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--; 
    TNCOH--; 
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
(MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
(MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= WEANFDATE) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;  
}
else{
    animal->DIE = 0;
}
if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT=0;
    tempcow->LDAY=0;
    tempcow->MCMFC=0;
    tempcow->MCSNF=0;
    tempcow->YN=0;
    tempcow->YEN=0;
    tempcow->CIDN=0;
    tempcow->CBIRTHDATE=0;
    tempcow->FWEAN++;
    CowsFailWean++;  
}
else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE)<= WEANFDATE) && (animal->REPLA==1) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        REPLR++;
        DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) &&
(MCDIE <= 0.00105525) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE > 101) && ( (animal->AGE + animal->BIRTHDATE) <= WEANFDATE) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else{
    animal->DIE = 0;
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
}

537
CowsFailWean++;
}
}
else if ( (animal->SEX == 2) && ((animal->AGE +
animal->BIRTHDATE) <= WEANFDATE) ){
  if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
  }
else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
  }
else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
  }
else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
  }
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
  }
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMS++;
    TNCOH--;
  }
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE
<= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE > 10) && (MCDIE <= 0.0002166) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE > 41) && (MCDIE <= 0.00006365) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= WEANFDATE) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else{
    animal->DIE = 0;  
}
if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);  
tempcow->LACT = 0;  
tempcow->LDAY = 0;  
tempcow->MCMFC=0;  
tempcow->MCSNF=0;  
tempcow->YN=0;  
tempcow->YEN=0;  
tempcow->CIDN=0;  
tempcow->CBIRTHDATE=0;  
tempcow->FWEAN++;
    CowsFailWean++;  
}
} else if ((animal->CALFSEAS==2) && ((WEANFDATE - (animal->AGE + animal->BIRTHDATE)) < 0 ) && (animal->SEX<3) && (EZBREED ==1) && (EZCALF != 1)){
    if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (animal->REPLA==0 ) ){
      if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMH++;
        NumAvailRepla--;
        TNCOH--;
      }
    }
} else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  DNUM++;
  DNUMH++;
  NumAvailRepla--;
  TNCOH--;
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  DNUM++;
  DNUMH++;
  NumAvailRepla--;
  TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  DNUM++;
  DNUMH++;
  NumAvailRepla--;
  TNCOH--;
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  DNUM++;
  DNUMH++;
  NumAvailRepla--;
  TNCOH--;
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  DNUM++;
DNUMH++; 
NumAvailRepla--; 
TNCOH--; 
}

    else if ( (animal->AGE > 6) && (animal->AGE <= 9) && 
        (MCDIE <= 0.00105525) ){
            animal->DIE = 1; 
            animal->AGED = animal->AGE; 
            DNUM++; 
            DNUMH++; 
            NumAvailRepla--; 
            TNCOH--; 
    }

    else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
            animal->DIE = 1; 
            animal->AGED = animal->AGE; 
            DNUM++; 
            DNUMH++; 
            NumAvailRepla--; 
            TNCOH--; 
    }

    else if ( (animal->AGE > 10) && (animal->AGE <= 41) && 
        (MCDIE <= 0.0002166) ){
            animal->DIE = 1; 
            animal->AGED = animal->AGE; 
            DNUM++; 
            DNUMH++; 
            NumAvailRepla--; 
            TNCOH--; 
    }

    else if ( (animal->AGE > 41) && (animal->AGE <= 101) && 
        (MCDIE <= 0.00006365) ){
            animal->DIE = 1; 
            animal->AGED = animal->AGE; 
            DNUM++; 
            DNUMH++; 
            NumAvailRepla--; 
            TNCOH--; 
    }

    else if ( (animal->AGE > 101) && ((animal->AGE + 
        animal->BIRTHDATE)<= (WEANFDATE + 365)) && (MCDIE <= 
        0.000034144) ){
            animal->DIE = 1; 
            animal->AGED = animal->AGE; 
            DNUM++; 
            DNUMH++; 
            NumAvailRepla--; 
            TNCOH--; 
    }

    else{
            animal->DIE = 0;
if (animal->DIE == 1) {
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
else if ((animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (animal->REPLA==1)) {
if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
} else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
} else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE
  <= 0.00105525) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
} else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
} else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
  (MCDIE <= 0.0002166) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
} else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
  (MCDIE <= 0.00006365) ){
  animal->DIE = 1;
  animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
} else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<=
  (WEANFDATE + 365)) && (MCDIE <= 0.000034144)
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else{
animal->DIE = 0;
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
else if ( (animal->SEX == 2) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) ){
if ( (animal->AGE == 1) && (MCDIE <= 0.0311) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--;
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.00322) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--;
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.00179) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00183) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++; 
DNUMS++; 
TNCOH--; 
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1; 
    animal->AGED = animal->AGE; 
    DNUM++; 
    DNUMS++; 
    TNCOH--; 
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1; 
    animal->AGED = animal->AGE; 
    DNUM++; 
    DNUMS++; 
    TNCOH--; 
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE 
< 0.00105525) ){
    animal->DIE = 1; 
    animal->AGED = animal->AGE; 
    DNUM++; 
    DNUMS++; 
    TNCOH--; 
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1; 
    animal->AGED = animal->AGE; 
    DNUM++; 
    DNUMS++; 
    TNCOH--; 
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && 
(MCDIE <= 0.0002166) ){
    animal->DIE = 1; 
    animal->AGED = animal->AGE; 
    DNUM++; 
    DNUMS++; 
    TNCOH--; 
}
else if ( (animal->AGE > 41) && (animal->AGE <= 101) && 
(MCDIE <= 0.00006365) ){
    animal->DIE = 1; 
    animal->AGED = animal->AGE; 
    DNUM++; 
    DNUMS++; 
    TNCOH--; 
}
else if ( (animal->AGE > 101) && ((animal->AGE + animal- 
>BIRTHDATE)<= (WEANFDATE + 365)) && (MCDIE <= 0.000034144) 
}
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--;
}
else{
    animal->DIE = 0;
}
if (animal->DIE == 1){
    tempcow = GetAnimal(animal->MIDN);
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
    tempcow->MCMFC = 0;
    tempcow->MCSNF = 0;
    tempcow->YN = 0;
    tempcow->YEN = 0;
    tempcow->CIDN = 0;
    tempcow->CBIRTHDATE = 0;
    tempcow->FWEAN = ++;
    CowsFailWean++;
}

    else if ( (animal->CALFSEAS == 2) && ((WEANFDATE - (animal->AGE + animal->BIRTHDATE)) >= 0 ) && (animal->SEX < 3) &&
        ((EZBREED != 1) || (EZBREED == 1)) && (EZCALF == 1)){
        if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= WEANFDATE) && (animal->REPLA == 0) ){
            if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
                }
            else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
                animal->DIE = 1;
                animal->AGED = animal->AGE;
                DNUM++;
                DNUMH++;
                NumAvailRepla--;
                TNCOH--;
                }
            else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--; 
TNCOH--; 
}
TNCOH--; }

else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
(MCDIE <= 0.00006365) )
{ animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM+;
DNUMH+;
NumAvailRepla--;
TNCOH--; }

else if ( (animal->AGE > 101) && ((animal->AGE +
animal->BIRTHDATE) <= WEANFDATE) && (MCDIE <= 0.000034144) )
{ animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM+;
DNUMH+;
NumAvailRepla--;
TNCOH--; }
else
{ animal->DIE = 0; }

if (animal->DIE == 1)
{ tempcow = GetAnimal(animal->MIDN);
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}

else if ( (animal->SEX == 0) && ((animal->AGE + animal->
BIRTHDATE) <= WEANFDATE) && (animal->REPLA==1) )
{ if ( (animal->AGE == 1) && (MCDIE <= 0.02143) )
{ animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR+;
DNUM+;
DNUMH+;
TNCOH--; }
else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) )
{ animal->DIE = 1;
animal->AGED = animal->AGE;

else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ) {
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    REPLR++;
    DNUM++;
    DNUMH++;
    TNCOH--;
}
DNUMH++;  
TNCOH--;  
}  
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&  
(MCDIE <= 0.0002166) )
    {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}  
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&  
(MCDIE <= 0.00006365) )
    {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}  
else if ( (animal->AGE > 101) && (animal->AGE + 
(animal->BIRTHDATE)<= WEANFDATE) && (MCDIE <= 0.000034144) )
    {  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
REPLR++;  
DNUM++;  
DNUMH++;  
TNCOH--;  
}  
else{  
animal->DIE = 0;  
}  
if (animal->DIE == 1)
    {  
tempcow = GetAnimal(animal->MIDN);  
tempcow->LACT = 0;  
tempcow->LDAY = 0;  
tempcow->MCMFC=0;  
tempcow->MCSNF=0;  
tempcow->YN=0;  
tempcow->YEN=0;  
tempcow->CIDN=0;  
tempcow->CBIRTHDATE=0;  
tempcow->FWEAN++;  
CowsFailWean++;  
}  
else if ( (animal->SEX == 2) && ((animal->AGE + 
animal->BIRTHDATE)<= WEANFDATE) )
    {  
if ( (animal->AGE == 1) && (MCDIE <= 0.02143) )
        {  
animal->DIE = 1;  
}  
}
animal->AGED = animal->AGE;
DNUM++;
DNUMS++;
TNCOH--;
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNMS++;
    TNCOH--;
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNMS++;
    TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNMS++;
    TNCOH--;
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNMS++;
    TNCOH--;
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNMS++;
    TNCOH--;
}
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE
    <= 0.00105525) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNMS++;
    TNCOH--;
}
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
}
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
(MCDIE <= 0.0002166) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
(MCDIE <= 0.00006365) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= WEANFDATE) && (MCDIE <= 0.000034144) ){  
animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else{  
animal->DIE = 0;  }
if (animal->DIE == 1){  
tempcow = GetAnimal(animal->MIDN);  
tempcow->LACT = 0;  
tempcow->LDAY = 0;  
tempcow->MCMFC=0;  
tempcow->MCSNF=0;  
tempcow->YN=0;  
tempcow->YEN=0;  
tempcow->CIDN=0;  
tempcow->CBIRTHDATE=0;  
tempcow->FWEAN++;  
CowsFailWean++;  }
}
}
else if ((animal->CALFSEAS==2) && ((WEANFDATE - (animal->AGE + animal->BIRTHDATE)) < 0 ) && (animal->SEX<3) &&
((EZBREED==1) || (EZBREED !=1)) && (EZCALF == 1)){  
if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (animal->REPLA==0) ){  

552
if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
} else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
} else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
} else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
} else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
} else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--;
} else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
    animal->DIE = 1;
animal->AGED = animal->AGE;
DNUM++;
DNUMH++;
NumAvailRepla--;
TNCOH--; 
} else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--; 
} else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&
    (MCDIE <= 0.0002166 )){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--; 
} else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&
    (MCDIE <= 0.00006365) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--; 
} else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (MCDIE <= 0.000034144) ){
    animal->DIE = 1;
    animal->AGED = animal->AGE;
    DNUM++;
    DNUMH++;
    NumAvailRepla--;
    TNCOH--; 
} else{
    animal->DIE = 0;
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
}
else if ( (animal->SEX == 0) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) && (animal->REPLA==1) ){
if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){
  animal->DIE = 1;
animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
}
else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
  animal->DIE = 1;
animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
}
else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){
  animal->DIE = 1;
animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
}
else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){
  animal->DIE = 1;
animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
}
else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
  animal->DIE = 1;
animal->AGED = animal->AGE;
  REPLR++;
  DNUM++;
  DNUMH++;
  TNCOH--;
}
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
} else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE <= 0.00105525) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
} else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
} else if ( (animal->AGE > 10) && (animal->AGE <= 41) && (MCDIE <= 0.0002166) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
} else if ( (animal->AGE > 41) && (animal->AGE <= 101) && (MCDIE <= 0.00006365) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
} else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= (WEANFDATE + 365)) && (MCDIE <= 0.000034144) ){
animal->DIE = 1;
animal->AGED = animal->AGE;
REPLR++;
DNUM++;
DNUMH++;
TNCOH--;
}
else{
    animal->DIE = 0;
}
}
if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->FWEAN++;
CowsFailWean++;
}
else if ( (animal->SEX == 2) && ((animal->AGE + animal->BIRTHDATE) <= (WEANFDATE + 365)) ){
    if ( (animal->AGE == 1) && (MCDIE <= 0.02143) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 2) && (MCDIE <= 0.00271) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 3) && (MCDIE <= 0.00156) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 4) && (MCDIE <= 0.00158) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
        DNUM++;
        DNUMS++;
        TNCOH--;
    }
    else if ( (animal->AGE == 5) && (MCDIE <= 0.00134) ){
        animal->DIE = 1;
        animal->AGED = animal->AGE;
    }
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 6) && (MCDIE <= 0.000804) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 6) && (animal->AGE <= 9) && (MCDIE  
<= 0.00105525) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE == 10) && (MCDIE <= 0.000737) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 10) && (animal->AGE <= 41) &&  
(MCDIE <= 0.0002166) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 41) && (animal->AGE <= 101) &&  
(MCDIE <= 0.00006365) ){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= (WEANFDATE + 365)) && (MCDIE <= 0.000034144)  
){
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}  
else if ( (animal->AGE > 101) && ((animal->AGE + animal->BIRTHDATE)<= (WEANFDATE + 365)) && (MCDIE <= 0.000034144)  
}{
    animal->DIE = 1;  
animal->AGED = animal->AGE;  
DNUM++;  
DNUMS++;  
TNCOH--;  
}
else{
    animal->DIE = 0;
}

if (animal->DIE == 1){
tempcow = GetAnimal(animal->MIDN);
tempcow->LACT = 0;
tempcow->LDAY = 0;
tempcow->MCMFC = 0;
tempcow->MCSNF = 0;
tempcow->YN = 0;
tempcow->YEN = 0;
tempcow->CIDN = 0;
tempcow->CBIRTHDATE = 0;
tempcow->FWEAN++;
CowsFailWean++;
}

    else if ( (animal->SEX == 4) && (animal->REPLA == 0) && (animal->AGE >= animal->WEAN) && (animal->AGE <= (3 * 365)) ){
        if (MCDIE <= 0.000054795){
animal->DIE = 1;
animal->AGED = animal->AGE;
NSHFC--;
DNUMHS++;
TNSCOH--;
}

    else if ( (animal->SEX == 4) && (animal->REPLA == 1) && (animal->AGE >= animal->WEAN) && (animal->AGE <= (3 * 365)) ){
        if (MCDIE <= 0.000054795){
animal->DIE = 1;
animal->AGED = animal->AGE;
NSHFC--;
DNUMHS++;
REPLR++;
REPLF--;
TNSCOH--;
}

    else if ( (animal->SEX == 5) && (animal->AGE >= animal->WEAN) && (animal->AGE <= 3*365) ){
        if (MCDIE <= 0.000054795){
animal->DIE = 1;
animal->AGED = animal->AGE;
DNUMSS++;
NSSTC--;
TNSCOH--;
}
else if ( (animal->SEX == 3) && (animal->CULL==0) )
    if (MCDIE <= 0.000041096)
        animal->DIE = 1;
        animal->AGED=animal->AGE;
        REPLR++;
        DNUMF++;
        if ( (animal->PREG==1) && (animal->CIDN==0))
            DeleteAnimal(animal->FIDN);
            animal->CSEX=0;
            animal->FIDN=0;
            PNUM--;  
    else if ((animal->PREG==1) && (animal->CIDN>0))
        DeleteAnimal(animal->FIDN);
        animal->CSEX=0;
        animal->FIDN=0;
        PNUM--;  
        tempcow = GetAnimal(animal->CIDN);
        tempcow->SellCalf=1;
        if ( (tempcow->SEX==0) && (tempcow->REPLA==1) && (tempcow->
          SellCalf==1) )
            REPLR++;
            TNCOH--; 
            BreedingReplacement--;  
    else if ( (tempcow->SEX==0) && (tempcow->REPLA==0) &&
          (tempcow->SellCalf==1) )
            NumAvailRepla--; 
            TNCOH--;  
    }  
else if ( (animal->PREG==0) && (animal->CIDN>0))
    tempcow = GetAnimal(animal->CIDN);
    tempcow->SellCalf=1;
    if ( (tempcow->SEX==0) && (tempcow->REPLA==1) && (tempcow->
          SellCalf==1) )
        REPLR++;
        TNCOH--; 
        BreedingReplacement--;  
    else if ( (tempcow->SEX==0) && (tempcow->REPLA==0) &&
          (tempcow->SellCalf==1) )
            NumAvailRepla--; 
            TNCOH--;  
    }  
else if ( (animal->SEX == 3) && (animal->CULL==0) )
    if (MCDIE <= 0.000041096){
animal->DIE = 1;
animal->AGED=animal->AGE;
DNUMF++;
if ((animal->PREG==1) && (animal->CIDN==0)){
    DeleteAnimal(animal->FIDN);
animal->CSEX=0;
animal->FIDN=0;
PNUM--;
}
else if ((animal->PREG==1) && (animal->CIDN>0)){
    DeleteAnimal(animal->FIDN);
animal->CSEX=0;
animal->FIDN=0;
PNUM--;
tempcow = GetAnimal(animal->CIDN);
tempcow->SellCalf=1;
if ( (tempcow->SEX==0) && (tempcow->REPLA==1) && (tempcow->
    >SellCalf==1) ){
    REPLR++;
    TNCOH--;
    BreedingReplacement--;
}
else if ( (tempcow->SEX==0) && (tempcow->REPLA==0) &&
     (tempcow->SellCalf==1) ){
    NumAvailRepla--;
    TNCOH--;
}
else if ((animal->PREG==0) && (animal->CIDN>0)){
    tempcow = GetAnimal(animal->CIDN);
tempcow->SellCalf=1;
if ( (tempcow->SEX==0) && (tempcow->REPLA==1) && (tempcow->
    >SellCalf==1) ){
    REPLR++;
    TNCOH--;
    BreedingReplacement--;
}
else if ( (tempcow->SEX==0) && (tempcow->REPLA==0) &&
     (tempcow->SellCalf==1) ){
    NumAvailRepla--;
    TNCOH--;
}
}

void Cow::ReproStats(){
    if (currentDATE == 365){
        if (NCTB>0){
            ACDAT = TCDAT / (double) NCTB;
        }
    }
}
AvGestLen=TotGestLen/(double)NCTB;
}
else{
ACDAT=0;
}
if ((COWS>0) && (BUYP==0) && (SpringCalfSeasn==1) && (NumBreedSeasns==1))
CRATE=(CNUMB/(double) CalfCows) * 100;
WRATE=(TNCW/(double) CalfCows) * 100;
CULRAT=(CLNUM/(double) CowsElig) * 100;
CNRAT = (CONCE/(double) (CowsElig)) * 100;
if (CNUMB>0)
AVGBWT=TBWT/CNUMB;
}else{
AVGBWT=0;
}
if (CNUMBH>0)
AVGBWTH=TBWTH/CNUMBH;
else{
AVGBWTH=0;
}
if (CNUMBS>0)
AVGBWTS=TBWTS/CNUMBS;
else{
AVGBWTS=0;
}
else if ((COWS>0) && (BUYP==0) && (FallCalfSeasn==1) &&
(NumBreedSeasns<=2))
WRATE=(TNCW/(double) CalfCows1) * 100;
CULRAT=(CLNUM/(double) CowsElig1) * 100;
CNRAT = (CONCE1/(double) (CowsElig1)) * 100;
if (ACDAT + AvGestLen >365)
CRATE=(CNUMB/(double) CalfCows1) * 100;
else if (ACDAT + AvGestLen <=365)
CRATE=(CNUMB/(double) CowsElig1) * 100;
if (CNUMB>0)
AVGBWT=TBWT/CNUMB;
}else{
AVGBWT=0;
}
if (CNUMBH>0)
AVGBWTH=TBWTH/CNUMBH;
else{
AVGBWTH=0;
}
else if (CNUMBS>0){
    AVGBWTS=TBWTS/CNUMBS;
}
else{
    AVGBWTS=0;
}
else if ((COWS>0) && (BUYP==1) && (SpringCalfSeasn == 1) &&
    (NumBreedSeasns==1)){
    CRATE=(CNUMB/(double) COWS) * 100;
    WRATE=(TNCW/(double) COWS) * 100;
    CULRAT=(CLNUM/(double) COWS) * 100;
    CNRAT = (CONCE/(double) (CowsElig)) * 100;
    if (CNUMB>0){
        AVGBWTS=TBWTS/CNUMBS;
    }
    else{
        AVGBWTS=0;
    }
    if (CNUMBH>0){
        AVGBWTH=TBWTH/CNUMBH;
    }
    else{
        AVGBWTH=0;
    }
    if (CNUMBS>0){
        AVGBWTS=TBWTS/CNUMBS;
    }
    else{
        AVGBWTS=0;
    }
    else if ((COWS>0) && (BUYP==1) && (FallCalfSeasn == 1) &&
    (NumBreedSeasns<=2)){
    WRATE=(TNCW/(double) COWS) * 100;
    CULRAT=(CLNUM/(double) COWS) * 100;
    CNRAT = (CONCE1/(double) (CowsElig1)) * 100;
    if (ACDAT + AvGestLen >365){
        CRATE=(CNUMB/(double) CalfCows1) * 100;
    }
    else if (ACDAT + AvGestLen <=365){
        CRATE=(CNUMB/(double) CowsElig1) * 100;
    }
    if (CNUMB>0){
        AVGBWTS=TBWTS/CNUMBS;
    }
    else{
        AVGBWTS=0;
    }
if (CNUMBH>0){
    AVGBWTH=TBWTH/CNUMBH;
} else{
    AVGBWTH=0;
}

if (CNUMBS>0){
    AVGBWTS=TBWTS/CNUMBS;
} else{
    AVGBWTS=0;
}

else{
    CRATE=0;
    WRATE=0;
    CULRAT=0;
    CNRAT =0;
}

if (((CONCE>0) && (SpringCalfSeasn==1) && (NumBreed-Seasns==1))
    AvServConcp=Servi/(double)CONCE;
} else if (((CONCE1>0) && (FallCalfSeasn==1) && (NumBreed- Seasns<=2))
    AvServConcp=Serv1/(double)CONCE1;
} else{
    AvServConcp=0;
}

if (TNCW>0){
   AWNWT=(WNWT/ (double) TNCW) * 100;
} else{
   AWNWT=0;
}

if (FCALF>0){
    AVFWEANAGE=(TFWEANAGE/(double) FCALF);
} else{
    AVFWEANAGE=0;
}

if (SCALF>0){
    AVSWEANAGE=(TSWEANAGE/(double) SCALF);
} else{
    AVSWEANAGE=0;
}

}
void Cow::Wean(AnimalRecord *animal) {
if ((animal->CALFSEAS==1) && (currentDATE==WEANSDATE) &&
(WEANSDATE - (animal->BIRTHDATE + (animal->AGE -1))) >= 0) ){
if ( (animal->SEX == 2) && (((animal->AGE - 1)+ animal->BIRTHDATE) == WEANSDATE) ){
if (animal->WEAN==0){
TWSS += animal->CDWT;
NSTC++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
animal->WEAN=animal->AGE;
TSWEANAGE+=animal->WEAN;
SCALF++;}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
}
else if ( (animal->SEX == 0) && (animal->REPLA == 0) &&
(((animal->AGE - 1) + animal->BIRTHDATE) == WEANSDATE) ){
if (animal->WEAN==0){
TWHS += animal->CDWT;
NHFC++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
animal->WEAN=animal->AGE;
TSWEANAGE+=animal->WEAN;
SCALF++;
}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
}
else if ( (animal->SEX == 0) && (animal->REPLA == 1) &&
(((animal->AGE - 1)+ animal->BIRTHDATE)== WEANSDATE) ){
if (animal->WEAN==0){
REPLF++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
animal->WEAN=animal->AGE;
TSWEANAGE+=animal->WEAN;
SCALF++;}
if (animal->WEANDATE==0){

}
animal->WEANDATE=currentDATE;
}
}
else if ( (animal->CalfSEAS==1) && (currentDATE==WEANSDATE)
&& ((WEANSDATE - (animal->BIRTHDATE + (animal->AGE - 1))) < 0) )
{
  if ( (animal->SEX == 2) && (((animal->AGE - 1) + animal->
                  BIRTHDATE) == (WEANSDATE + 365)) )
    {
      if (animal->WEAN==0)
      
        TWSS += animal->CDWT;
        NSTC++;
        WNWT += animal->CDWT;
        animal->YN=0;
        animal->YEN=0;
        animal->WEAN=animal->AGE;
        TSWEANAGE+=animal->WEAN;
        SCALF++;
    } else if (animal->WEANDATE==0)
    {animal->WEANDATE=currentDATE; 
    }
  }
else if ( (animal->SEX == 0) && (animal->REPLA == 0) &&
(( (animal->AGE - 1) + animal->BIRTHDATE) == (WEANSDATE +
365)) )
{
  if (animal->WEAN==0)
  
    TWHS += animal->CDWT;
    NHFC++;
    WNWT += animal->CDWT;
    animal->YN=0;
    animal->YEN=0;
    animal->WEAN=animal->AGE;
    TSWEANAGE+=animal->WEAN;
    SCALF++;
  } else if (animal->WEANDATE==0)
  {animal->WEANDATE=currentDATE; 
  }
else if ( (animal->SEX == 0) && (animal->REPLA == 1) &&
(( (animal->AGE - 1) + animal->BIRTHDATE) == (WEANSDATE +
365)) )
{
  if (animal->WEAN==0)
  REPLF++;
  TNCW++;
  WNWT += animal->CDWT;
  animal->YN=0;
  animal->YEN=0;


animal->WEAN=animal->AGE;
TSWEANAGE+=animal->WEAN;
SCALF++;
}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
}
else if ( (animal->CalfSEAS==2) && (currentDATE==WEANFDATE) && ((WEANFDATE - (animal->BIRTHDATE + (animal->AGE - 1))) >=0) ){ if ( (animal->SEX == 2) && (((animal->AGE - 1) + animal->BIRTHDATE) == WEANFDATE) ){ if (animal->WEAN==0){ TWSS += animal->CDWT;
NSTC++;
TNCW++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
FCALF++;
animal->WEAN=animal->AGE;
TFWEANAGE+=animal->WEAN;
} if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
}
else if ( (animal->SEX == 0) && (animal->REPLA == 0) && (((animal->AGE - 1) + animal->BIRTHDATE) == WEANFDATE) ){ if (animal->WEAN==0){ TWHS += animal->CDWT;
NHFC++;
TNCW++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
FCALF++;
animal->WEAN=animal->AGE;
TFWEANAGE+=animal->WEAN;
} if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
}
else if ( (animal->SEX == 0) && (animal->REPLA == 1) && (((animal->AGE - 1) + animal->BIRTHDATE) == WEANFDATE) ){ if (animal->WEAN==0){ REPLF++;
TNCW++;
}
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
FCALF++;
animal->WEAN=animal->AGE;
TFWEANAGE+=animal->WEAN;
}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}
else if ( (animal->SEX == 2) && (((animal->AGE - 1) + animal->BIRTHDATE) == (WEANFDATE + 365)) ){
if (animal->WEAN==0){
TWSS += animal->CDWT;
NSTC++;
TNCW++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
FCALF++;
animal->WEAN=animal->AGE;
TFWEANAGE+=animal->WEAN;
}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}

else if ( (animal->SEX == 0) && (animal->REPLA == 0) && (((animal->AGE - 1) + animal->BIRTHDATE) == (WEANFDATE + 365)) ){
if (animal->WEAN==0){
TWHS += animal->CDWT;
NHFC++;
TNCW++;
WNWT += animal->CDWT;
animal->YN=0;
animal->YEN=0;
FCALF++;
animal->WEAN=animal->AGE;
TFWEANAGE+=animal->WEAN;
}
if (animal->WEANDATE==0){
animal->WEANDATE=currentDATE;
}

else if ( (animal->SEX == 0) && (animal->REPLA == 1) && (((animal->AGE - 1) + animal->BIRTHDATE) == (WEANFDATE +}
if (animal->WEAN==0){
    REPLF++;
    TNCW++;
    WNWT += animal->CDWT;
    animal->YN=0;
    animal->YEN=0;
    FCALF++;
    animal->WEAN=animal->AGE;
    TFWEANAGE+=animal->WEAN;
}
if (animal->WEANDATE==0){
    animal->WEANDATE=currentDATE;
}
}
if( (currentDATE==365) && ((animal->AGE/7)>=56) && ((animal->SEX ==3) || ((animal->SEX==0) && (animal->REPLA==1)) || ((animal->SEX==4) && (animal->REPLA==1))) ){
    TOTALAGE+=animal->AGE;
}
if( (currentDATE==365) && ((animal->SEX ==3) || ((animal->SEX==0) && (animal->REPLA==1)) || ((animal->SEX==4) && (animal->REPLA==1))) ){
    if (animal->AGE <365){
        AgeBreeders[0]++;
        NumCowsYend++;
    }
    else if ((animal->AGE >=365) && (animal->AGE <730)){
        AgeBreeders[1]++;
        NumCowsYend++;
    }
    else if ((animal->AGE >=730) && (animal->AGE <1095)){
        AgeBreeders[2]++;
        NumCowsYend++;
    }
    else if ((animal->AGE >=1095) && (animal->AGE <1460)){
        AgeBreeders[3]++;
        NumCowsYend++;
    }
    else if ((animal->AGE >=1460) && (animal->AGE <1825)){
        AgeBreeders[4]++;
        NumCowsYend++;
    }
    else if ((animal->AGE >=1825) && (animal->AGE <2190)){
        AgeBreeders[5]++;
        NumCowsYend++;
    }
    else if ((animal->AGE >=2190) && (animal->AGE <2555)){
        AgeBreeders[6]++;
        NumCowsYend++;
}
else if ((animal->AGE >=2555) && (animal->AGE <2920))
    AgeBreeders[7]++;
NumCowsYend++;
}
else if ((animal->AGE >=2920) && (animal->AGE <3285))
    AgeBreeders[8]++;
NumCowsYend++;
}
else if ((animal->AGE >=3285) && (animal->AGE <3650))
    AgeBreeders[9]++;
NumCowsYend++;
}
else if ((animal->AGE >=3650) && (animal->AGE <4015))
    AgeBreeders[10]++;
NumCowsYend++;
}
else if ((animal->AGE >=4015) && (animal->AGE <4380))
NumCowsYend++;
}
else if ((animal->AGE >=4380) && (animal->AGE<5110))
    AgeBreeders[12]++;
NumCowsYend++;
}
else if (animal->AGE>=5110)
    AgeBreeders[14]++;
NumCowsYend++;
}
else if ( (SpringCalfSeasn== 1) && (animal->UNPREG==0) &&
    (animal->PREG==0) && (currentDATE==LDBS) && (animal->CULL == 0) &&
    ((animal->SEX ==3) && (animal->AGE>=56/7)) )
{
    if (animal->BCS == 1)
        BCScore[1]++;
    else if (animal->BCS ==2)
        BCScore[2]++;
    else if (animal->BCS ==3)
        BCScore[3]++;
    else if (animal->BCS ==4)
        BCScore[4]++;
    else if (animal->BCS ==5){
else if (animal->BCS ==5){
  BCScore[5]++;
}
else if (animal->BCS ==6){
  BCScore[6]++;
}
else if (animal->BCS ==7){
  BCScore[7]++;
}
else if (animal->BCS ==8){
  BCScore[8]++;
}
else if (animal->BCS ==9){
  BCScore[9]++;
}
else if ( (FallCalfSeasn== 1) && (LDBS2<=365) && (animal->UNPREG==0) && (animal->PREG==0) && (currentDATE==LDBS2) && (animal->CULL==0) && ((animal->SEX ==3)) ){
  if (animal->BCS == 1){
    BCScore[1]++;
  }
  else if (animal->BCS ==2){
    BCScore[2]++;
  }
  else if (animal->BCS ==3){
    BCScore[3]++;
  }
  else if (animal->BCS ==4){
    BCScore[4]++;
  }
  else if (animal->BCS ==5){
    BCScore[5]++;
  }
  else if (animal->BCS ==6){
    BCScore[6]++;
  }
  else if (animal->BCS ==7){
    BCScore[7]++;
  }
  else if (animal->BCS ==8){
    BCScore[8]++;
  }
  else if (animal->BCS ==9){
    BCScore[9]++;
  }
}
else if ( (FallCalfSeasn== 1) && (LDBS2>365) && (animal->UNPREG==0) && (animal->PREG==0) && (currentDATE==LDBS2-365) && (animal->CULL==0) && ((animal->SEX ==3)) ){

if (animal->BCS == 1){
    BCScore[1]++;
}
else if (animal->BCS == 2){
    BCScore[2]++;
}
else if (animal->BCS == 3){
    BCScore[3]++;
}
else if (animal->BCS == 4){
    BCScore[4]++;
}
else if (animal->BCS == 5){
    BCScore[5]++;
}
else if (animal->BCS == 6){
    BCScore[6]++;
}
else if (animal->BCS == 7){
    BCScore[7]++;
}
else if (animal->BCS == 8){
    BCScore[8]++;
}
else if (animal->BCS == 9){
    BCScore[9]++;
}
}

if ((currentDATE==365) && (animal->SEX>3) && (animal->REPLA==0)){
    NumStkrYEnd++;
}
}

void Cow::Sterility(AnimalRecord *animal){

double UNPREGH = .032;

double UNPREGC = .05;

double randomnumber=rand();

double MCSTER = (randomnumber/(double) RAND_MAX);

if ((animal->SEX==3) && (animal->UNPREG == 0) && (animal->AGE>=56/7)){
    if ( ((currentDATE == (FDBHS - 1)) || (currentDATE ==
         (FDBHS2 - 1)))
         && (animal->PARIE == 0) && (MCSTER <= UNPREGH)
         && (animal->UNPREG==0) && (animal->PREG==0))
        {animal->UNPREG=1;
        }
    else if ( (animal->PREG == 1) && (animal->DPREG >= animal->PGEST) &&
        (animal->PARIE >= 0) && (animal->DYSTO == 1) && (MCSTER <=
        UNPREGC)){
animal->UNPREG=1;
}
else{
    animal->UNPREG=0;
}
}

void Cow::ShrinkAnimal(AnimalRecord *animal){
    double PSBWMK=0;
    double SRWK = 478;
    if ((animal->SEX == 2) || (animal->SEX == 5)){
        animal->SBWTK = animal->CDWTK * .96;
        PSBWMK = ((animal->PMSWT*.453599) * .96);
        animal->EBWTK = (animal->SBWTK * .891);
        animal->EQSBW=(animal->SBWTK * (SRWK/PSBWMK));
        animal->EQEBW=animal->EQSBW * .891;
    }
    else if (((animal->SEX==0) || (animal->SEX == 3) ||
        (animal->SEX == 4)) && (animal->PREG == 0)){
        animal->SBWTK = animal->CDWTK * .96;
        PSBWMK = (animal->PMWT*.453599) * .96;
        animal->EBWTK = animal->SBWTK * .891;
        animal->EQSBW=animal->SBWTK * (SRWK/PSBWMK);
        animal->EQEBW=animal->EQSBW * .891;
    }
    else if ((animal->SEX==3) && (animal->PREG == 1)){
        animal->SBWTK = animal->CDWTK * .96;
        PSBWMK = (animal->PMWT*.453599) * .96;
        animal->EBWTK = (animal->SBWTK - (animal->GUWTK))* .891;
        animal->EQSBW=(animal->SBWTK - (animal->GUWTK)) *
        (SRWK/PSBWMK);
        animal->EQEBW=animal->EQSBW * .891;
    }
}

void Cow::CullCows(AnimalRecord *animal){
    if ( (YEAR == -7) && (FallCalfSeasn==1) && (CULPO==0)){
        animal->SEAOP=0;
        TSTOP=--;
    }
    else if ( ((YEAR== -7) || (YEAR== -6)) && (SpringCalfSeasn==1)
        && (currentDATE==LDBS+PTEST) && (CULPO>0) ){
        animal->FAIL=0;
        animal->SEAOP=0;
        animal->FWEAN=0;
    }
    else if ( ((YEAR== -7) || (YEAR== -6)) && (FallCalfSeasn==1)
        && (currentDATE==LDBS2+PTEST-365) && (CULPO>0) &

animal->FAIL=0;
animal->SEAOP=0;
animal->FWEAN=0;
}
else if ( ((YEAR==-7) || (YEAR==-6)) && (FallCalfSeasn==1) && (currentDATE==LDBS2+PTEST) && (CULPO>0) && (LDBS2+PTEST<=365)) {
animal->FAIL=0;
animal->SEAOP=0;
animal->FWEAN=0;
}
if ( (animal->CULL==0) && (animal->SEX==3) ){
   if (CULPO==0){
      if ( (SpringCalfSeasn==1) && (currentDATE == LDBS+PTEST) ){
         animal->CULL=1;
         REPLR++;
         CulledCows++;
      }
   } else if ( FallCalfSeasn==1) {
      if ( ((LDBS2 + PTEST) <=365) && (currentDATE == LDBS2 + PTEST) ) {
         if (animal->SEAOP > 0){
            animal->CULL=1;
            REPLR++;
            CulledCows++;
         }
      }
   } else if ( ((LDBS2 + PTEST) > 365) && (currentDATE == LDBS2 + PTEST - 365) ) {
      if (animal->SEAOP > 0){
         animal->CULL=1;
         REPLR++;
         CulledCows++;
      }
   }
}
else if ( (CULPO == 1) && ((animal->SEAOP + animal->FAIL + animal->FWEAN) >= 1) ){
   animal->CULL=1;
   REPLR++;
   CulledCows++;
}
else if ( (CULPO == 2) && ((animal->SEAOP + animal->FAIL + animal->FWEAN) >= 2) ){
   animal->CULL=1;
   REPLR++;
}
CulledCows++;

} else if ( (CULPO == 3) && ((animal->SEAOP + animal->FAIL + animal->FWEAN) >= 3) ) {
    animal->CULL=1;
    REPLR++;
    CulledCows++;
}

if( (CULPO<=3) && (animal->AGE) >= (CULAG * 365) && (animal->CULL==0) ) {
    animal->CULL=1;
    REPLR++;
    CulledCows++;
}

if ( (CULPO<=3) && (animal->BCS==1) && (animal->CULL==0) ) {
    animal->CULL=1;
    REPLR++;
    CulledCows++;
}

if ( (animal->CULL==1) && (animal->PREG==1) ) {
    DeleteAnimal(animal->FIDN);
    PNUM--;
}

if ( (animal->SEX==3) && (animal->CULL==1) && (animal->LACT==0) ) {
    CLNUM++;
}

AnimalRecord* Cow::GetSex(int sex, int startPos){
    AnimalRecord *head;
    head = GetAnimal(startPos);
    while (head != NULL){
        if (head->SEX == sex){
            return head;
        }
        head=head->next;
    }
    return NULL;
}

void Cow::ResetCowVariables(){
    AWWT=0;
    CRATE=0;
    CLNUM=0;
    CNUMB=0;
    CNUMBH=0;
    CNUMBS=0;
    CNRAT=0;
    CULRAT=0;
}
DNUM=0;
DNUMF=0;
DNUMH=0;
DNUMHS=0;
DNUMR=0;
DNUMS=0;
DNUMSS=0;
Dystocia2=0;
Dystocia3=0;
Dystocia4=0;
NCTB=0;
NHFC1=NHFC;
NSTC1=NSTC;
NHFC=0;
NSTC=0;
PHEF=0;
POHEF=0;
REPLF=0;
SELEC=0;
TCDAT=0;
TNCW=0;
TSTOP=0;
TWHS1=TWHS;
TWSS1=TWSS;
TWHS=0;
TWSS=0;
WNWT=0;
TBWT=0;
AVGBWT=0;
TBWTS=0;
AVGBWTS=0;
TBWTH=0;
AVGBWTH=0;
TFWEANAGE=0;
TSWEANAGE=0;
WNWT=0;
WRATE=0;
AVFWEANAGE=0;
AVSWEANAGE=0;
FCALF=0;
SCALF=0;
CalfCows=CowsElig;
CalfCows1=CowsElig1;
AvGestLen=0;
TotGestLen=0;
CowsFail=0;
CowsFailWean=0;
CowsAbort=0;
EligCowsOpen=0;
BCScore[1]=0;
BCScore[2]=0;
void Cow::SBBreeders(AnimalRecord *animal){
    int i=0;
    double MCHOW=0;  /*A stochastic variable used to compute
    double MCAGE = GetStochasticNumber (35, 570);
    double MCOAGE = GetStochasticNumber (27, 254);
    if (NumBreedSeasns==1){
        if (SpringCalfSeasn==1){
            MCHOW = GetStochasticNumber ( ((.8 * (SEAS-(SEAS/2)))/3.9),
                (((SEAS/2) + 25) + PTEST) );
        }
    }
    else if (FallCalfSeasn==1){
        MCHOW = GetStochasticNumber ( ((.8 * (SEAS-(SEAS/2)))/3.9),
                (((SEAS2/2) + 25) + PTEST) );
    }
    else if (NumBreedSeasns==2){
        if ( ((currentState - PTEST)==LDBS) ){
            double MCHOW = GetStochasticNumber ( ((.8 * (SEAS-(SEAS/2)))3.9),
                ((SEAS/2) + PTEST) );
        }
        else if ( ((currentState - PTEST)==LDBS2) && ((LDBS2 +
            PTEST) <=365) ){
            double MCHOW = GetStochasticNumber ( ((.8 * (SEAS-
(SEAS/2))/3.9), ((SEAS2/2) + PTEST) ;
}
else if ((currentDATE - PTEST)==(LDB2-365)) &
((LDB2 + PTEST) >365) ){
double MCHOW = GetStochasticNumber ( ((.8 * (SEAS-
(SEAS/2))/3.9), ((SEAS2/2) + PTEST) );
}
if ((TEST==0) && (REPLR>0) ) {
if ( (BUYP == 1) && (TSTOP > 0) ){
while (TSTOP> 0){
AnimalRecord *tempmother = new AnimalRecord;
AnimalRecord *tempcow = new AnimalRecord;
tempmother->ID=0;
if (YEAR<1){
tempmother->DAMBREED=HBREED;
tempmother->PURE=PURE;
tempmother->SIREBREED=HBREED;
}
else if ((YEAR>=1) && (BuyPure == 0)){
tempmother->DAMBREED=BuyBreed + 1;
tempmother->PURE=PURE;
tempmother->SIREBREED=BuyBreed + 1;
}
else if ((YEAR>=1) && (BuyPure == 1)){
tempmother->PURE=PURE + 1;
tempmother->DAMBREED=BuyBreed + 1;
tempmother->SIREBREED=0;
}
if (tempmother->DAMBREED==1){
tempmother->PMWT=GetStochasticNumber(139.96, 1156);
tempmother->BWT =GetStochasticNumber(9.91 , 80.9);
tempmother->ACBWT =80.9;
tempmother->PADG=GetStochasticNumber(.16 , 1.87);
tempmother->UADG=GetStochasticNumber((antiADG/11.685),
antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempmother->PMSWT=GetStochasticNumber(26.0 , 969);
tempmother->PGAVG=285.4;
tempmother->PGSD=6.02;
tempmother->PGEST = GetStochasticNumber( 6.02, 285.4);
}
else if (tempmother->DAMBREED==2){
tempmother->PMWT=GetStochasticNumber(143.84, 1188);
tempmother->BWT = GetStochasticNumber(10.56, 86.2);
tempmother->ACBWT = 86.2;
tempmother->PADG = GetStochasticNumber(0.145, 1.70);
tempmother->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempmother->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempmother->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempmother->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempmother->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempmother->PMSWT = GetStochasticNumber(27.85, 1038);
tempmother->PGAVG = 288.6;
tempmother->PGSD = 6.09;
tempmother->PGEST = GetStochasticNumber(6.09, 288.6);
}
else if (tempmother->DAMBREED == 3){
tempmother->PMWT = GetStochasticNumber(143.84, 1188);
tempmother->BWT = GetStochasticNumber(10.56, 86.2);
tempmother->ACBWT = 86.2;
tempmother->PADG = GetStochasticNumber(0.145, 1.70);
tempmother->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempmother->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempmother->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempmother->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempmother->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempmother->PMSWT = GetStochasticNumber(27.85, 1038);
tempmother->PGAVG = 288.6;
tempmother->PGSD = 6.09;
tempmother->PGEST = GetStochasticNumber(6.09, 288.6);
}
else if (tempmother->DAMBREED == 4){
tempmother->PMWT = GetStochasticNumber(143.84, 1188);
tempmother->BWT = GetStochasticNumber(10.51, 85.8);
tempmother->ACBWT = 85.8;
tempmother->PADG = GetStochasticNumber(0.154, 1.8);
tempmother->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempmother->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempmother->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempmother->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempmother->PMSWT=GetStochasticNumber(26.51 , 988);
tempmother->PGAVG=288.6;
tempmother->PGSD=6.09;
tempmother->PGEST = GetStochasticNumber( 6.09, 288.6);
}
else if (tempmother->DAMBREED==5){
tempmother->PMWT=GetStochasticNumber(168.42, 1391);
tempmother->BWT =GetStochasticNumber(11.1 , 90.6);
tempmother->ACBWT =90.6;
tempmother->PADG=GetStochasticNumber(.158 , 1.85);
tempmother->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempmother->PMSWT=GetStochasticNumber(33.218 , 1238);
tempmother->PGAVG=285.9;
tempmother->PGSD=6.03;
tempmother->PGEST = GetStochasticNumber( 6.03, 285.9);
}
else if (tempmother->DAMBREED==6){
tempmother->PMWT=GetStochasticNumber(189.24, 1563);
tempmother->BWT =GetStochasticNumber(10.94 , 89.3);
tempmother->ACBWT =89.3;
tempmother->PADG=GetStochasticNumber(.157 , 1.83);
tempmother->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempmother->PMSWT=GetStochasticNumber(36.223 , 1350);
tempmother->PGAVG=287.5;
tempmother->PGSD=6.07;
tempmother->PGEST = GetStochasticNumber( 6.07, 287.5);
}
else if (tempmother->DAMBREED==7){
tempmother->PMWT=GetStochasticNumber(156.67, 1294);
tempmother->BWT =GetStochasticNumber(10.54 , 86.0);
tempmother->ACBWT =86.0;
tempmother->PADG=GetStochasticNumber(.16, 1.87);
tempmother->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempmother->PMSWT=GetStochasticNumber(31.527, 1175);
tempmother->PGAVG=286.3;
tempmother->PGSD=6.04;
tempmother->PGEST = GetStochasticNumber( 6.04, 286.3);
} else if (tempmother->DAMBREED==8){
tempmother->PMWT=GetStochasticNumber(146.62, 1211);
tempmother->BWT =GetStochasticNumber(10.51, 85.8);
tempmother->ACBWT =85.8;
tempmother->PADG=GetStochasticNumber(.151, 1.76);
tempmother->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempmother->PMSWT=GetStochasticNumber(29.864, 1113);
tempmother->PGAVG=288.1;
tempmother->PGSD=6.08;
tempmother->PGEST = GetStochasticNumber( 6.08, 288.1);
} else if (tempmother->DAMBREED==9){
tempmother->PMWT=GetStochasticNumber(184.16, 1521);
tempmother->BWT =GetStochasticNumber(11.1, 90.6);
tempmother->ACBWT =90.6;
tempmother->PADG=GetStochasticNumber(.154, 1.81);
tempmother->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempmother->PMSWT=GetStochasticNumber(33.218 , 1238);
tempmother->PGAVG=285.4;
tempmother->PGSD=6.02;
tempmother->PGEST = GetStochasticNumber( 6.02, 285.4);
}  
else if (tempmother->DAMBREED==10){
tempmother->PMWT=GetStochasticNumber(149.17, 1232);
tempmother->BWT =GetStochasticNumber(10.58 , 86.4);
tempmother->ACBWT =86.4;
tempmother->PADG=GetStochasticNumber(.151 , 1.76);
tempmother->UADG=GetStochasticNumber((antiADG/11.685),
antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempmother->PMSWT=GetStochasticNumber(28.844 , 1075);
tempmother->PGAVG=286.0;
tempmother->PGSD=6.03;
tempmother->PGEST = GetStochasticNumber( 6.03, 286.0);
}  
else if (tempmother->DAMBREED==11){
tempmother->PMWT=GetStochasticNumber(156.91, 1296);
tempmother->BWT =GetStochasticNumber(10.89 , 88.9);
tempmother->ACBWT =88.9;
tempmother->PADG=GetStochasticNumber(.157 , 1.83);
tempmother->UADG=GetStochasticNumber((antiADG/11.685),
antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempmother->PMSWT=GetStochasticNumber(32.547 , 1213);
tempmother->PGAVG=286.3;
tempmother->PGSD=6.04;
tempmother->PGEST = GetStochasticNumber( 6.04, 286.3);
}  
else if (tempmother->DAMBREED==12){
tempmother->PMWT=GetStochasticNumber(137.3, 1134);
tempmother->BWT =GetStochasticNumber(10.13 , 82.7);
tempmother->ACBWT =82.7;
tempmother->PADG=GetStochasticNumber(.154 , 1.81);
tempmother->UADG=GetStochasticNumber((antiADG/11.685),
antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempmother->PMSWT=GetStochasticNumber(39.245 , 1291);
tempmother->PGAVG=286.2;
tempmother->PGSD=6.05;
tempmother->PGEST = GetStochasticNumber( 6.05, 286.2);
antiADG);
tempmother->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempmother->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempmother->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempmother->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempmother->PMSWT=GetStochasticNumber(28.495 , 1062);
tempmother->PGAVG=287.1;
tempmother->PGSD=6.06;
tempmother->PGEST = GetStochasticNumber( 6.06, 288.6);
}
tempmother->AGE = MCAGE;
tempmother->AGED =0;
tempmother->BREED=tempmother->DAMBREED;
tempmother->CBWT=0;
if ( (tempmother->BWT + (tempmother->PADG * tempmother->AGE))<= (tempmother->PMWT) ){
tempmother->CDWT = ((tempmother->BWT) + (tempmother->PADG *
tempmother->AGE)) ;
}
else{
tempmother->CDWT = tempmother->PMWT;
}
tempmother->CDWTK=(tempmother->CDWT*.453599);
tempmother->SBWTK=((tempmother->CDWT)* .96);
tempmother->EBWTK=tempmother->SBWTK * .891;
tempmother->EQSBW=((tempmother->SBWTK) * (478/(tempmother->PMWT*.453599)));
tempmother->EQEBW=tempmother->EQSBW * .891;
tempmother->GUWTK=0;
tempmother->BIRTHDATE=0;
tempmother->Calfseas=0;
tempmother->WEAN=0;
tempmother->WEANDATE=0;
tempmother->WEANWT=0;
tempmother->Dhetbwt=0;
tempmother->Dhetadg=0;
tempmother->Dhetmwt=0;
tempmother->MHETADG=0;
tempmother->MHETBWT=0;
tempmother->Seaop=0;
tempmother->Seaop1=0;
tempmother->Seaop2=0;
tempmother->BCS=5;
tempmother->COLR=COLR;
tempmother->SEX=3;
tempmother->DIE=0;
tempmother->REPLA=0;
tempmother->SOUR=1;
tempmother->CULL=0;
tempmother->CADAT=0;
tempmother->IMPLANT=0;
tempmother->DETEC=0;
tempmother->CUCYC=0;
tempmother->PREG=1;
tempmother->PARIE = 0;
tempmother->DPREG= MCHOW;
tempmother->DYSTO=0;
tempmother->FAIL=0;
tempmother->FWEAN=0;
tempmother->PUBER=0;
tempmother->INELEG=0;
tempmother->Puberty=1;
tempmother->UNPREG=0;
tempmother->PPI=0;
tempmother->LACT=0;
tempmother->LDAY=0;
tempmother->FIDN=0;
tempmother->CSEX=0;
tempmother->CBWT=0;
tempmother->SERVI=0;
tempmother->PAST=1;
tempmother->CIDN=0;
tempmother->CBIRTHDATE=0;
tempmother->MIDN=0;
tempmother->MCMFC=0;
tempmother->MCSNF=0;
tempmother->CNDAT = (currentDATE - tempmother->DPREG);
tempmother->Fgeneration=0;
tempmother->GDAMBREED=0;
tempmother->GGDAMBREED=0;
tempmother->GGGDAMBREED=0;
tempmother->GSIREBREED=0;
tempmother->GGSIREBREED=0;
tempmother->GGGSIREBREED=0;
tempmother->SellCalf=0;
tempmother->YN=0;
tempmother->YEN=0;
tempmother->YFAIL=0;
tempmother->next=NULL;
AddAnimal(tempmother);
PHEF++;
REPLR--;
BreedingCows++;
TSTOP--;
PNUM++;
if ( (currentDATE - (tempmother->DPREG)) < 0 ){
tempmother->CNDAT=(currentDATE - (tempmother->DPREG) +365);
}
else{
    tempmother->CNDAT=(currentDATE - (tempmother->DPREG));
}
tempcow->DAMBREED = tempmother->BREED;
tempcow->BREED = tempcow->DAMBREED;
if (BuyPure==0){
tempcow->SIREBREED = tempcow->DAMBREED;
}
else{
tempcow->SIREBREED = tempcow->DAMBREED;
}
BirthAnimal(tempcow, tempmother);
delete tempmother;
tempmother=GetAnimal(tempcow->MIDN);
tempmother->FIDN=tempcow->ID;
tempmother->CBWT=tempcow->BWT;
tempmother->CSEX=tempcow->SEX;
for (i=1; i<=tempmother->DPREG; i++){
tempmother->GUWTK+= (((animal->CBWT) * .453599) * (.3656 -
(.000523 * (i))) * (pow(2.71881828, ( (.02 * (i)) - (.0000143
* (pow((i),2)))))/1000  ));
}
tempmother->GUWTK= (((((tempmother->CBWT) * .453599) /40.699) * .06973 * (pow(2.718281828, ((.0323 * (tempmother-
>DPREG)) - (.0000275 * (tempmother->DPREG) * (tempmother-
>DPREG))))/1000  ))/;)
tempmother->SBWTK=tempmother->SBWTK + tempmother->GUWTK;
tempmother->CDWTK=(tempmother->SBWTK/.96);
tempmother->CDWT=tempmother->CDWTK *2.20459;
tempmother->EBWTK=tempmother->SBWTK *.891;
tempmother->EQSBW=( (tempmother->SBWTK) * (478/((tempmother-
>PMWT*.453599) * .96)));
tempmother->EQEBW=tempmother->EQSBW *.891;
delete tempcow;
}
}
else if ( (BUYP == 0) && (TSTOP > 0) && (BUYRC==1) ){while (TSTOP> 0){
AnimalRecord *tempcow = new AnimalRecord;
tempcow->ID=0;
tempcow->SEX=3;
if (YEAR<1){
tempcow->DAMBREED=HBREED;
tempcow->BREED=HBREED;
tempcow->PURE=PURE;
tempcow->SIREBREED=HBREED;
tempcow->Fgeneration=0;
}
else if ((YEAR>=1) && (BuyPure == 0)){
tempcow->DAMBREED=BuyBreed + 1;
tempcow->BREED=BuyBreed + 1;
tempcow->PURE=PURE;
tempcow->SIREBREED=BuyBreed + 1;
tempcow->Fgeneration=0;
}
else if ((YEAR>=1) && (BuyPure == 1)){
tempcow->PURE=PURE + 1;
tempcow->DAMBREED=BuyBreed + 1;
tempcow->BREED=BuyBreed + 1;
tempcow->SIREBREED=0;
tempcow->Fgeneration=0;
}
if (tempcow->DAMBREED==1){
tempcow->PMWT=GetStochasticNumber(139.96, 1156);
tempcow->BWT =GetStochasticNumber(9.91 , 80.9);
tempcow->ACBWT =80.9;
tempcow->PADG=GetStochasticNumber(.16 , 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.0 , 969);
tempcow->PGAVERAGE=285.4;
tempcow->PGSD=6.02;
}
else if (tempcow->DAMBREED==2){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
tempcow->ACBWT =86.2;
tempcow->PADG=GetStochasticNumber(.145 , 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
tempcow->PGAVERAGE=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==3){

586
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
tempcow->ACBWT =86.2;
tempcow->PADG=GetStochasticNumber(.145 , 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==4){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.51 , 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.154 , 1.8);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.51 , 988);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==5){
tempcow->PMWT=GetStochasticNumber(168.42, 1391);
tempcow->BWT =GetStochasticNumber(11.1 , 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.158 , 1.85);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218, 1238);
tempcow->PGAVG=285.9;
tempcow->PGSD=6.03;
} else if (tempcow->DAMBREED==6){
tempcow->PMWT=GetStochasticNumber(189.24, 1563);
tempcow->BWT =GetStochasticNumber(10.94, 89.3);
tempcow->ACBWT =89.3;
tempcow->PADG=GetStochasticNumber(.157, 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(36.223, 1350);
tempcow->PGAVG=287.5;
tempcow->PGSD=6.07;
} else if (tempcow->DAMBREED==7){
tempcow->PMWT=GetStochasticNumber(156.67, 1294);
tempcow->BWT =GetStochasticNumber(10.54, 86.0);
tempcow->ACBWT =86.0;
tempcow->PADG=GetStochasticNumber(.16, 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(31.527, 1175);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
} else if (tempcow->DAMBREED==8){
tempcow->PMWT=GetStochasticNumber(146.62, 1211);
tempcow->BWT =GetStochasticNumber(10.51, 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.151, 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(29.864, 1113);
tempcow->PGAVG=288.1;
tempcow->PGSD=6.08;
}
else if (tempcow->DAMBREED==9){
tempcow->PMWT=GetStochasticNumber(184.16, 1521);
tempcow->BWT =GetStochasticNumber(11.1, 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.154, 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218, 1238);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
}
else if (tempcow->DAMBREED==10){
tempcow->PMWT=GetStochasticNumber(149.17, 1232);
tempcow->BWT =GetStochasticNumber(10.58, 86.4);
tempcow->ACBWT =86.4;
tempcow->PADG=GetStochasticNumber(.151, 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844, 1075);
tempcow->PGAVG=286.0;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==11){
tempcow->PMWT=GetStochasticNumber(156.91, 1296);
tempcow->BWT = GetStochasticNumber(10.89, 88.9);
tempcow->ACBWT = 88.9;
tempcow->PADG = GetStochasticNumber(.157, 1.83);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
    tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(32.547, 1213);
tempcow->PGAVG = 286.3;
tempcow->PGSD = 6.04;

else if (tempcow->DAMBREED == 12){
tempcow->PMWT = GetStochasticNumber(137.3, 1134);
tempcow->BWT = GetStochasticNumber(10.13, 82.7);
tempcow->ACBWT = 82.7;
tempcow->PADG = GetStochasticNumber(.154, 1.81);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(28.495, 1062);
tempcow->PGAVG = 287.1;
tempcow->PGSD = 6.06;
}
tempcow->AGE = MCOAGE;
tempcow->AGED = 0;
if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)) > (tempcow->PMWT) ){
tempcow->CDWT = tempcow->PMWT;
}
else if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)) <= (tempcow->PMWT) ){
tempcow->CDWT = ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE));
}
tempcow->SEAOP = 0;
tempcow->SEAOP1 = 0;
tempcow->SEAOP2 = 0;
tempcow->BCS = 5;
tempcow->COLR=COLR;
tempcow->DIE=0;
tempcow->REPLA=1;
tempcow->SOUR=1;
tempcow->CULL=0;
tempcow->CADAT=0;
tempcow->CNDAT=0;
tempcow->DETEC=0;
tempcow->PGEST=0;
tempcow->IMPLANT=0;
tempcow->CUCYC=0;
tempcow->PREG=0;
tempcow->PARTE = 0;
tempcow->DPREG=0;
tempcow->DYSTO=0;
tempcow->FAIL=0;
tempcow->FWEAN=0;
tempcow->PUBER=0;
tempcow->INELIG=1;
tempcow->Puberty=0;
tempcow->UNPREG=0;
tempcow->PPI=0;
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->FIDN=0;
tempcow->CSEX=0;
tempcow->GUWTK=0;
tempcow->CBWT=0;
tempcow->SERVI=0;
tempcow->PAST=1;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->MIDN=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->DAMBREED=tempcow->DAMBREED;
tempcow->GDAMBREED=0;
tempcow->GGDAMBREED=0;
tempcow->GGGDAMBREED=0;
tempcow->GSIREBREED=0;
tempcow->GGSIREBREED=0;
tempcow->GGGSIREBREED=0;
tempcow->CDWTK=(tempcow->CDWT*.453599);
tempcow->SBWTK=((tempcow->CDWT*.453599) * .96);
tempcow->EBWTK=tempcow->SBWTK * .891;
tempcow->EQSBW=((tempcow->SBWTK) * (478/(tempcow->PMWT*.453599)));
tempcow->EQEBW=tempcow->EQSBW * .891;
tempcow->SellCalf=0;
tempcow->SIREBREED=0;
tempcow->BIRTHDATE=0;
tempcow->CALFSEAS=0;
tempcow->WEAN=0;
tempcow->WEANDATE=0;
tempcow->WEANWT=0;
tempcow->DHETBWT=0;
tempcow->DHETAGD=0;
tempcow->DHETMWT=0;
tempcow->MHETAGD=0;
tempcow->MHETBWT=0;
tempcow->YFAIL=0;
tempcow->next=NULL;
AddAnimal(tempcow);
POHEF++;
REPLR--;
BreedingCows++;
TSTOP--;
delete tempcow;
}
else if ( ( ( (BUYP == 1) && (TSTOP == 0) ) || (BUYP == 0) )
  && (REPLR>0) && (NumAvailRepla>0) && (SelectWIHerd==1) ){
    AnimalRecord *animalPointer;
    int StartPosition;
    animalPointer = GetHeadAnimal();
    animalPointer = animalPointer->next;
    StartPosition = animalPointer->ID;
    while ( (REPLR>0) && (NumAvailRepla>0) ){
        animalPointer=GetSex(0, StartPosition);
        while  ( (animalPointer != NULL) && (REPLR>0) && (NumAvail -
            Repla>0) ){
            if ( (animalPointer->REPLA == 0) && (animalPointer->AGE>0)
            ){
                REPLR--;
                NumAvailRepla--;
                animalPointer->REPLA = 1;
                SELEC++;
                BreedingReplacement++;
            }
        animalPointer = animalPointer->next;
        if (animalPointer != NULL){
            StartPosition = animalPointer->ID;
        }
        animalPointer = GetSex(0, StartPosition);
    }
    }
else if ( ( ( (BUYP == 1) && (TSTOP == 0) ) || (BUYP == 0) )
  && (SelectWIHerd==1) && (NumAvailRepla==0) &&
  (BUYRC==1) && (REPLR>0) ){
while (REPLR>0){
    AnimalRecord *tempcow = new AnimalRecord;
    tempcow->ID=0;
    tempcow->SEX=3;
    if (YEAR<1){
        tempcow->DAMBREED=HBREED;
        tempcow->BREED=HBREED;
        tempcow->PURE=PURE;
        tempcow->SIREBREED=HBREED;
        tempcow->Fgeneration=0;
    }
    else if ((YEAR>=1) && (BuyPure == 0)){
        tempcow->DAMBREED=BuyBreed + 1;
        tempcow->BREED=BuyBreed + 1;
        tempcow->PURE=PURE;
        tempcow->SIREBREED=BuyBreed + 1;
        tempcow->Fgeneration=0;
    }
    else if ((YEAR>=1) && (BuyPure == 1)){
        tempcow->PURE=PURE + 1;
        tempcow->DAMBREED=BuyBreed + 1;
        tempcow->BREED=BuyBreed + 1;
        tempcow->SIREBREED=0;
        tempcow->Fgeneration=0;
    }
    if (tempcow->DAMBREED==1){
        tempcow->PMWT=GetStochasticNumber(139.96, 1156);
        tempcow->BWT =GetStochasticNumber(9.91  , 80.9);
        tempcow->ACBWT =80.9;
        tempcow->PADG=GetStochasticNumber(.16   , 1.87);
        tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
        tiADG);
        tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
        tiADG1);
        tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
        tiADG2);
        tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
        tiADG3);
        tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
        tiADG4);
        tempcow->PMSWT=GetStochasticNumber(26.0 , 969);
        tempcow->PGAVG=285.4;
        tempcow->PGSD=6.02;
    }else if (tempcow->DAMBREED==2){
        tempcow->PMWT=GetStochasticNumber(143.84, 1188);
        tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
        tempcow->ACBWT =86.2;
        tempcow->PADG=GetStochasticNumber(.145   , 1.70);
        tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
        tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);

else if (tempcow->DAMBREED==3){
    tempcow->PMWT=GetStochasticNumber(143.84, 1188);
    tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
    tempcow->ACBWT =86.2;
    tempcow->PADG=GetStochasticNumber(.145 , 1.70);
    tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
    tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
    tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
    tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
    tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
    tempcow->PGAVG=288.6;
    tempcow->PGSD=6.09;
}

else if (tempcow->DAMBREED==4){
    tempcow->PMWT=GetStochasticNumber(143.84, 1188);
    tempcow->BWT =GetStochasticNumber(10.51 , 85.8);
    tempcow->ACBWT =85.8;
    tempcow->PADG=GetStochasticNumber(.154 , 1.8 );
    tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
    tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
    tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
    tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
    tempcow->PMSWT=GetStochasticNumber(26.51 , 988);
    tempcow->PGAVG=288.6;
    tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==5){


tempcow->PMWT=GetStochasticNumber(168.42, 1391);
tempcow->BWT =GetStochasticNumber(11.1 , 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.158 , 1.85);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218 , 1238);
tempcow->PGAVG=285.9;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==6){
tempcow->PMWT=GetStochasticNumber(189.24, 1563);
tempcow->BWT =GetStochasticNumber(10.94 , 89.3);
tempcow->ACBWT =89.3;
tempcow->PADG=GetStochasticNumber(.157 , 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(36.223 , 1350);
tempcow->PGAVG=287.5;
tempcow->PGSD=6.07;
}
else if (tempcow->DAMBREED==7){
tempcow->PMWT=GetStochasticNumber(156.67, 1294);
tempcow->BWT =GetStochasticNumber(10.54 , 86.0);
tempcow->ACBWT =86.0;
tempcow->PADG=GetStochasticNumber(.16 , 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4),
tempcow->PMSWT=GetStochasticNumber(31.527, 1175);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==8){
tempcow->PMWT=GetStochasticNumber(146.62, 1211);
tempcow->BWT =GetStochasticNumber(10.51, 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(29.864, 1113);
tempcow->PGAVG=288.1;
tempcow->PGSD=6.08;
}
else if (tempcow->DAMBREED==9){
tempcow->PMWT=GetStochasticNumber(184.16, 1521);
tempcow->BWT =GetStochasticNumber(11.1, 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.154 , 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218, 1238);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
}
else if (tempcow->DAMBREED==10){
tempcow->PMWT=GetStochasticNumber(149.17, 1232);
tempcow->BWT =GetStochasticNumber(10.58, 86.4);
tempcow->ACBWT =86.4;
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844,1075);
tempcow->PGAVG=286.0;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==11){
tempcow->PMWT=GetStochasticNumber(156.91,1296);
tempcow->BWT =GetStochasticNumber(10.89,88.9);
tempcow->ACBWT =88.9;
tempcow->PADG=GetStochasticNumber(.157,1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(32.547,1213);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==12){
tempcow->PMWT=GetStochasticNumber(137.3,1134);
tempcow->BWT =GetStochasticNumber(10.13,82.7);
tempcow->ACBWT =82.7;
tempcow->PADG=GetStochasticNumber(.154,1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.495,1062);
tempcow->PGAVG=287.1;
tempcow->PGSD=6.06;
}
tempcow->AGE = MCOAGE;
tempcow->AGED = 0;
if ( ((tempcow->BWT + (tempcow->PADG * tempcow->AGE)) > (tempcow->PMWT)) ){
    tempcow->CDWT = tempcow->PMWT;
}
else if ( ((tempcow->BWT + (tempcow->PADG * tempcow->AGE)) <= (tempcow->PMWT)) ){
    tempcow->CDWT = ((tempcow->BWT + (tempcow->PADG * tempcow->AGE)));
}
    tempcow->SEAOP=0;
    tempcow->SEAOP1=0;
    tempcow->SEAOP2=0;
    tempcow->BCS=5;
    tempcow->COLR=COLR;
    tempcow->DIE=0;
    tempcow->REPLA=1;
    tempcow->SOUR=1;
    tempcow->CULL=0;
    tempcow->CADAT=0;
    tempcow->CNDAT=0;
    tempcow->DETEC=0;
    tempcow->PGEST=0;
    tempcow->IMPLANT=0;
    tempcow->CUCYC=0;
    tempcow->PREG=0;
    tempcow->PARIE = 0;
    tempcow->DPREG=0;
    tempcow->DYSTO=0;
    tempcow->FAIL=0;
    tempcow->FWEAN=0;
    tempcow->PUBER=0;
    tempcow->INELEG=1;
    tempcow->Puberty=0;
    tempcow->UNPREG=0;
    tempcow->PI=0;
    tempcow->LACT=0;
    tempcow->LDAY=0;
    tempcow->FIDN=0;
    tempcow->CSEX=0;
    tempcow->GUWTK=0;
    tempcow->CBWT=0;
    tempcow->SERVI=0;
    tempcow->PAST=1;
    tempcow->CIDN=0;
    tempcow->CBIRTHDATE=0;
    tempcow->MIDN=0;
    tempcow->MCMFC=0;
    tempcow->MCSNF=0;
    tempcow->YN=0;
    tempcow->YEN=0;
    tempcow->DAMBREED=tempcow->DAMBREED;
tempcow->GDAMBREED=0;
tempcow->GGDAMBREED=0;
tempcow->GGGDAMBREED=0;
tempcow->GSIREBREED=0;
tempcow->GGSIREBREED=0;
tempcow->GGGSIREBREED=0;
tempcow->CDWTK=(tempcow->CDWT*.453599);
tempcow->SBWTK=((tempcow->CDWT*.453599)* .96);
tempcow->EBWTK=tempcow->SBWTK * .891;
tempcow->EQSBW=((tempcow->SBWTK) * (478/(tempcow->PMWT*.453599)));
tempcow->EQEBW=tempcow->EQSBW * .891;
tempcow->SellCalf=0;
tempcow->SIREBREED=0;
tempcow->WEANWT=0;
tempcow->BIRTHDATE=0;
tempcow->CALFSEAS=0;
tempcow->WEAN=0;
tempcow->WEANDATE=0;
tempcow->DHETBWT=0;
tempcow->DHETADG=0;
tempcow->DHETMWT=0;
tempcow->MHETADG=0;
tempcow->MHETBWT=0;
tempcow->YFAIL=0;
tempcow->next=NULL;
AddAnimal(tempcow);
POHEF++;
REPLR--;
BreedingCows++;
delete tempcow;
}

else if ( (BUYP == 1) && (TSTOP == 0) && (BUYRC==1) && (SelectWlHerd==0) && (REPLR>0)){
while (REPLR> 0){
AnimalRecord *tempcow = new AnimalRecord;
tempcow->ID=0;
tempcow->SEX=3;
if (YEAR<1){
tempcow->DAMBREED=HBREED;
tempcow->BREED=HBREED;
tempcow->PURE=PURE;
tempcow->SIREBREED=HBREED;
tempcow->Fgeneration=0;
}
else if ((YEAR>=1) && (BuyPure == 0)){
tempcow->DAMBREED=BuyBreed + 1;
tempcow->BREED=BuyBreed + 1;
tempcow->PURE=PURE;
tempcow->SIREBREED=BuyBreed + 1;

599
tempcow->Fgeneration=0;
} 
else if ((YEAR>=1) && (BuyPure == 1)){
    tempcow->PURE=PURE + 1;
    tempcow->DAMBREED=BuyBreed + 1;
    tempcow->BREED=BuyBreed + 1;
    tempcow->SIREBREED=0;
    tempcow->Fgeneration=0;
}
if (tempcow->DAMBREED==1){
    tempcow->PMWT=GetStochasticNumber(139.96, 1156);
    tempcow->BWT =GetStochasticNumber(9.91 , 80.9);
    tempcow->ACBWT =80.9;
    tempcow->PADG=GetStochasticNumber(.16 , 1.87);
    tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
        tiADG);
    tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
        tiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
        tiADG2);
    tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
        tiADG3);
    tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
        tiADG4);
    tempcow->PMSWT=GetStochasticNumber(26.0 , 969);
    tempcow->PGAVG=285.4;
    tempcow->PGSD=6.02;
} 
else if (tempcow->DAMBREED==2){
    tempcow->PMWT=GetStochasticNumber(143.84, 1188);
    tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
    tempcow->ACBWT =86.2;
    tempcow->PADG=GetStochasticNumber(.145 , 1.70);
    tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
        tiADG);
    tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
        tiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
        tiADG2);
    tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
        tiADG3);
    tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
        tiADG4);
    tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
    tempcow->PGAVG=288.6;
    tempcow->PGSD=6.09;
} 
else if (tempcow->DAMBREED==3){
    tempcow->PMWT=GetStochasticNumber(143.84, 1188);
    tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
    tempcow->ACBWT =86.2;
tempcow->PADG=GetStochasticNumber(.145, 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85, 1038);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==4){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT=GetStochasticNumber(10.51, 85.8);
tempcow->ACBWT=85.8;
tempcow->PADG=GetStochasticNumber(.154, 1.8);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.51, 988);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==5){
tempcow->PMWT=GetStochasticNumber(168.42, 1391);
tempcow->BWT=GetStochasticNumber(11.1, 90.6);
tempcow->ACBWT=90.6;
tempcow->PADG=GetStochasticNumber(.158, 1.85);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218, 1238);
tempcow->PGAVG=285.9;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==6){
tempcow->PMWT=GetStochasticNumber(189.24, 1563);
tempcow->BWT =GetStochasticNumber(10.94, 89.3);
tempcow->ACBWT =89.3;
tempcow->PADG=GetStochasticNumber(.157, 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
tiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
tiADG4);
tempcow->PMSWT=GetStochasticNumber(36.223, 1350);
tempcow->PGAVG=287.5;
tempcow->PGSD=6.07;
}
else if (tempcow->DAMBREED==7){
tempcow->PMWT=GetStochasticNumber(156.67, 1294);
tempcow->BWT =GetStochasticNumber(10.54, 86.0);
tempcow->ACBWT =86.0;
tempcow->PADG=GetStochasticNumber(.16, 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
tiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
tiADG4);
tempcow->PMSWT=GetStochasticNumber(31.527, 1175);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==8){
tempcow->PMWT=GetStochasticNumber(146.62, 1211);
tempcow->BWT =GetStochasticNumber(10.51, 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.151, 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(29.864, 1113);
tempcow->PGAVG=288.1;
tempcow->PGSD=6.08;
}  
else if (tempcow->DAMBREED==9){
tempcow->PMWT=GetStochasticNumber(184.16, 1521);
tempcow->BWT =GetStochasticNumber(11.1, 90.6);  
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.154, 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218, 1238);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
}  
else if (tempcow->DAMBREED==10){
tempcow->PMWT=GetStochasticNumber(149.17, 1232);
tempcow->BWT =GetStochasticNumber(10.58, 86.4);  
tempcow->ACBWT =86.4;
tempcow->PADG=GetStochasticNumber(.151, 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844, 1075);
tempcow->PGAVG=286.0;
tempcow->PGSD=6.03;
}  
else if (tempcow->DAMBREED==11){
tempcow->PMWT=GetStochasticNumber(156.91, 1296);
tempcow->BWT =GetStochasticNumber(10.89, 88.9);  
tempcow->ACBWT =88.9;
tempcow->PADG=GetStochasticNumber(.157, 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(32.547, 1213);
tempcow->PGAIVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==12){
tempcow->PMWT=GetStochasticNumber(137.3, 1134);
tempcow->BWT =GetStochasticNumber(10.13, 82.7);
tempcow->ACBWT =82.7;
tempcow->PADG=GetStochasticNumber(.154, 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.495, 1062);
tempcow->PGAIVG=287.1;
tempcow->PGSD=6.06;
}
tempcow->AGE = MCOAGE;
tempcow->AGED = 0;
if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE))>(tempcow->PMWT) ){
tempcow->CDWT=tempcow->PMWT;
}
else if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE))<=(tempcow->PMWT) ){
tempcow->CDWT = ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE));
}
tempcow->SEAOP=0;
tempcow->SEAOP1=0;
tempcow->SEAOP2=0;
tempcow->BCS=5;
tempcow->COLR=COLR;
tempcow->DIE=0;
tempcow->REPLA=1;
tempcow->SOUR=1;
tempcow->CULL=0;
tempcow->CADAT=0;
tempcow->CNDAT=0;
tempcow->DETEC=0;
tempcow->PGEST=0;
tempcow->IMPLANT=0;
tempcow->CUCYC=0;
tempcow->PREG=0;
tempcow->PARIE=0;
tempcow->DPREG=0;
tempcow->DYSTO=0;
tempcow->FAIL=0;
tempcow->FWEAN=0;
tempcow->PUBER=0;
tempcow->INELIG=1;
tempcow->Puberty=0;
tempcow->UNPREG=0;
tempcow->PPI=0;
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->FIDN=0;
tempcow->CSEX=0;
tempcow->GUWTK=0;
tempcow->CBWT=0;
tempcow->SERVI=0;
tempcow->PAST=1;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->MIDN=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->DAMBREED=tempcow->DAMBREED;
tempcow->GDAMBREED=0;
tempcow->GGDAMBREED=0;
tempcow->GGGDAMBREED=0;
tempcow->GSIREBREED=0;
tempcow->GGSIREBREED=0;
tempcow->GGGSIREDREED=0;
tempcow->CDWTK=(tempcow->CDWT*.453599);
tempcow->SBWTK=((tempcow->CDWT*.453599)*.96);
tempcow->EBWTK=tempcow->SBWTK*.891;
tempcow->EQSBW=((tempcow->SBWTK)*(478/(tempcow->PMWT*.453599)));
tempcow->EQEBW=tempcow->EQSBW*.891;
tempcow->SellCalf=0;
tempcow->SIREBREED=0;
tempcow->BIRTHDATE=0;
tempcow->CALFSEAS=0;
tempcow->WEAN=0;
tempcow->WEANDATE=0;
tempcow->WEANWT=0;
tempcow->DHETBWT=0;
tempcow->DHETADG=0;
tempcow->DHETMWT=0;
tempcow->MHETADG=0;
tempcow->MHETBWT=0;
tempcow->YFAIL=0;
tempcow->next=NULL;
AddAnimal(tempcow);
POHEF++;
REPLR--;
BreedingCows++;
delete tempcow;
}
else if ( (BUYP == 1) && (TSTOP == 0) && (BUYRC==1) &&
(SelectW1Herd==0) && (REPLR>0)) {
while (REPLR> 0) {
AnimalRecord *tempcow = new AnimalRecord;
tempcow->ID=0;
tempcow->SEX=3;
if (YEAR<1) {
tempcow->DAMBREED=HBREED;
tempcow->BREED=HBREED;
tempcow->PURE=PURE;
tempcow->SIREBREED=HBREED;
tempcow->Fgeneration=0;
}
else if ((YEAR>=1) && (BuyPure == 0)) {
tempcow->DAMBREED=BuyBreed + 1;
tempcow->BREED=BuyBreed + 1;
tempcow->PURE=PURE;
tempcow->SIREBREED=BuyBreed + 1;
tempcow->Fgeneration=0;
}
else if ((YEAR>=1) && (BuyPure == 1)) {
tempcow->PURE=PURE + 1;
tempcow->DAMBREED=BuyBreed + 1;
tempcow->BREED=BuyBreed + 1;
tempcow->SIREBREED=0;
tempcow->Fgeneration=0;
}
if (tempcow->DAMBREED==1) {
tempcow->PMWT=GetStochasticNumber(139.96, 1156);
tempcow->BWT=GetStochasticNumber(9.91, 80.9);
tempcow->ACBWT=80.9;
tempcow->PADG=GetStochasticNumber(.16, 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.0 , 969);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
} else if (tempcow->DAMBREED==2){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
tempcow->ACBWT =86.2;
tempcow->PADG=GetStochasticNumber(.145 , 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
} else if (tempcow->DAMBREED==3){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
tempcow->ACBWT =86.2;
tempcow->PADG=GetStochasticNumber(.145 , 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
} else if (tempcow->DAMBREED==4){

tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.51 , 85.8);
tempcow->ACBWT = 85.8;
tempcow->PADG=GetStochasticNumber(.154 , 1.8);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.51 , 988);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==5){
tempcow->PMWT=GetStochasticNumber(168.42, 1391);
tempcow->BWT =GetStochasticNumber(11.1 , 90.6);
tempcow->ACBWT = 90.6;
tempcow->PADG=GetStochasticNumber(.158 , 1.85);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218 , 1238);
tempcow->PGAVG=285.9;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==6){
tempcow->PMWT=GetStochasticNumber(189.24, 1563);
tempcow->BWT =GetStochasticNumber(10.94 , 89.3);
tempcow->ACBWT = 89.3;
tempcow->PADG=GetStochasticNumber(.157 , 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(36.223 , 1350);
tempcow->PGAVG=287.5;
tempcow->PGSD=6.07;
}
else if (tempcow->DAMBREED==7){
tempcow->PMWT=GetStochasticNumber(156.67, 1294);
tempcow->BWT =GetStochasticNumber(10.54 , 86.0);
tempcow->ACBWT =86.0;
tempcow->PADG=GetStochasticNumber(.16 , 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-

antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
tiADG);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
tiADG);
tempcow->PMSWT=GetStochasticNumber(31.527 , 1175);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==8){
tempcow->PMWT=GetStochasticNumber(146.62, 1211);
tempcow->BWT =GetStochasticNumber(10.51 , 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-

antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
tiADG);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
tiADG);
tempcow->PMSWT=GetStochasticNumber(29.864 , 1113);
tempcow->PGAVG=288.1;
tempcow->PGSD=6.08;
}
else if (tempcow->DAMBREED==9){
tempcow->PMWT=GetStochasticNumber(184.16, 1521);
tempcow->BWT =GetStochasticNumber(11.1 , 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.154 , 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218, 1238);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
}
else if (tempcow->DAMBREED==10){
tempcow->PMWT=GetStochasticNumber(149.17, 1232);
tempcow->BWT =GetStochasticNumber(10.58, 86.4);
tempcow->ACBWT =86.4;
tempcow->PAGD=GetStochasticNumber(.151, 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844, 1075);
tempcow->PGAVG=286.0;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==11){
tempcow->PMWT=GetStochasticNumber(156.91, 1296);
tempcow->BWT =GetStochasticNumber(10.89, 88.9);
tempcow->ACBWT =88.9;
tempcow->PAGD=GetStochasticNumber(.157, 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(32.547, 1213);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==12){
tempcow->PMWT=GetStochasticNumber(137.3, 1134);
tempcow->BWT = GetStochasticNumber(10.13, 82.7);
tempcow->ACBWT = 82.7;
tempcow->PADG = GetStochasticNumber(.154, 1.81);
tempcow->UADG = GetStochasticNumber((antiADG / 11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1 / 11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2 / 11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3 / 11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4 / 11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(28.495, 1062);
tempcow->PGAVG = 287.1;
tempcow->PGSD = 6.06;
}
tempcow->AGE = MCOAGE;
tempcow->AGED = 0;
if (((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)) > (tempcow->PMWT)) {
tempcow->CDWT = tempcow->PMWT;
}
else if (((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)) <= (tempcow->PMWT)) {
tempcow->CDWT = ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE));
}
tempcow->SEAOP = 0;
tempcow->SEAOP1 = 0;
tempcow->SEAOP2 = 0;
tempcow->BCS = 5;
tempcow->COLR = COLR;
tempcow->DIE = 0;
tempcow->REPLA = 1;
tempcow->SOUR = 1;
tempcow->CULL = 0;
tempcow->CADAT = 0;
tempcow->CNDAT = 0;
tempcow->DETEC = 0;
tempcow->PGEST = 0;
tempcow->IMPLANT = 0;
tempcow->CUCYC = 0;
tempcow->PREG = 0;
tempcow->PARIE = 0;
tempcow->DPREG = 0;
tempcow->DYSTO = 0;
tempcow->FAIL = 0;
tempcow->FWEAN = 0;
tempcow->PUBER = 0;
tempcow->INELIG = 1;
tempcow->Puberty=0;
tempcow->UNPREG=0;
tempcow->PPI=0;
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->FIDN=0;
tempcow->CSEX=0;
tempcow->GWTK=0;
tempcow->CBWT=0;
tempcow->SERVI=0;
tempcow->PAST=1;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->MIDN=0;
tempcow->MCBIRTHDATE=0;
tempcow->MCBIRTHDATE=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->DAMBREED=tempcow->DAMBREED;
tempcow->GDAMBREED=0;
tempcow->GDAMBREED=0;
tempcow->GDAMBREED=0;
tempcow->GDAMBREED=0;
tempcow->GSIREBREED=0;
tempcow->GSIREBREED=0;
tempcow->GSIREBREED=0;
tempcow->GSIREBREED=0;
tempcow->GDAMBREED=0;
tempcow->GDAMBREED=0;
tempcow->GDAMBREED=0;
tempcow->GDAMBREED=0;
tempcow->CDWTK=(tempcow->CDWT*.453599);
tempcow->SBWTK=((tempcow->CDWT*.453599)* .96);
tempcow->EBWTK=tempcow->SBWTK * .891;
tempcow->EQSBW=((tempcow->SBWTK) * (478/(tempcow->PMWT*.453599)));
tempcow->EQEBW=tempcow->EQSBW * .891;
tempcow->SellCalf=0;
tempcow->SIREBREED=0;
tempcow->BIRTHDATE=0;
tempcow->CALFSEAS=0;
tempcow->WEAN=0;
tempcow->WEANDATE=0;
tempcow->WEANWT=0;
tempcow->DHETBWT=0;
tempcow->DHETADG=0;
tempcow->DHETMWT=0;
tempcow->MHETADG=0;
tempcow->MHETBWT=0;
tempcow->YFAIL=0;
tempcow->next=NULL;
AddAnimal(tempcow);
POHEF++;
REPLR--;
BreedingCows++;
delete tempcow;
else if ( (BUYP == 0) && (TSTOP == 0) && (BUYRC==1) && 
(SelectW1Herd==0) && (REPLR>0)) {
    while (REPLR> 0) {
        AnimalRecord *tempcow = new AnimalRecord;
        tempcow->ID=0;
        tempcow->SEX=3;
        if (YEAR<1) {
            tempcow->DAMBREED=HBREED;
            tempcow->BREED=HBREED;
            tempcow->PURE=PURE;
            tempcow->SIREBREED=HBREED;
            tempcow->Fgeneration=0;
        } else if ((YEAR>=1) && (BuyPure == 0)) {
            tempcow->DAMBREED=BuyBreed + 1;
            tempcow->BREED=BuyBreed + 1;
            tempcow->PURE=PURE;
            tempcow->SIREBREED=BuyBreed + 1;
            tempcow->Fgeneration=0;
        } else if ((YEAR>=1) && (BuyPure == 1)) {
            tempcow->PURE=PURE + 1;
            tempcow->DAMBREED=BuyBreed + 1;
            tempcow->BREED=BuyBreed + 1;
            tempcow->SIREBREED=0;
            tempcow->Fgeneration=0;
        } if (tempcow->DAMBREED==1) {
            tempcow->PMWT=GetStochasticNumber(139.96, 1156);
            tempcow->BWT =GetStochasticNumber(9.91 , 80.9);
            tempcow->ACBWT =80.9;
            tempcow->PADG=GetStochasticNumber(.16 , 1.87);
            tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
            tiADG);
            tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
            tiADG1);
            tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
            tiADG2);
            tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
            tiADG3);
            tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
            tiADG4);
            tempcow->PMSWT=GetStochasticNumber(26.0 , 969);
            tempcow->PGAVG=285.4;
            tempcow->PGSD=6.02;
        } else if (tempcow->DAMBREED==2) {
            tempcow->PMWT=GetStochasticNumber(143.84, 1188);
            tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
            tempcow->ACBWT =86.2;
tempcow->PADG = GetStochasticNumber(.145, 1.70);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(27.85, 1038);
tempcow->PGAVG = 288.6;
tempcow->PGSD = 6.09;
}
else if (tempcow->DAMBREED == 3){
tempcow->PMWT = GetStochasticNumber(143.84, 1188);
tempcow->BWT = GetStochasticNumber(10.56, 86.2);
tempcow->ACBWT = 86.2;
tempcow->PADG = GetStochasticNumber(.145, 1.70);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(27.85, 1038);
tempcow->PGAVG = 288.6;
tempcow->PGSD = 6.09;
}
else if (tempcow->DAMBREED == 4){
tempcow->PMWT = GetStochasticNumber(143.84, 1188);
tempcow->BWT = GetStochasticNumber(10.51, 85.8);
tempcow->ACBWT = 85.8;
tempcow->PADG = GetStochasticNumber(.154, 1.8);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(26.51, 988);
tempcow->PGAVG = 288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==5){
tempcow->PMWT=GetStochasticNumber(168.42, 1391);
tempcow->BWT =GetStochasticNumber(11.1 , 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.158 , 1.85);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218 , 1238);
tempcow->PGAVG=285.9;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==6){
tempcow->PMWT=GetStochasticNumber(189.24, 1563);
tempcow->BWT =GetStochasticNumber(10.94 , 89.3);
tempcow->ACBWT =89.3;
tempcow->PADG=GetStochasticNumber(.157 , 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(36.223 , 1350);
tempcow->PGAVG=287.5;
tempcow->PGSD=6.07;
}
else if (tempcow->DAMBREED==7){
tempcow->PMWT=GetStochasticNumber(156.67, 1294);
tempcow->BWT =GetStochasticNumber(10.54 , 86.0);
tempcow->ACBWT =86.0;
tempcow->PADG=GetStochasticNumber(.16 , 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(31.527, 1175);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==8){
tempcow->PMWT=GetStochasticNumber(146.62, 1211);
tempcow->BWT =GetStochasticNumber(10.51, 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
tiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
tiADG4);
tempcow->PMSWT=GetStochasticNumber(29.864 , 1113);
tempcow->PGAVG=288.1;
tempcow->PGSD=6.08;
}
else if (tempcow->DAMBREED==9){
tempcow->PMWT=GetStochasticNumber(184.16, 1521);
tempcow->BWT =GetStochasticNumber(11.1, 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.154 , 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), an-
tiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), an-
tiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), an-
tiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), an-
tiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218 , 1238);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
}
else if (tempcow->DAMBREED==10){
tempcow->PMWT=GetStochasticNumber(149.17, 1232);
tempcow->BWT =GetStochasticNumber(10.58, 86.4);
tempcow->ACBWT =86.4;
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844, 1075);
tempcow->PGA aviation=286.0;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==11){
tempcow->PMWT=GetStochasticNumber(156.91, 1296);
tempcow->BWT =GetStochasticNumber(10.89, 88.9);
tempcow->ACBWT =88.9;
tempcow->PAGD=GetStochasticNumber(.157, 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844, 1075);
tempcow->PGA aviation=286.0;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==12){
tempcow->PMWT=GetStochasticNumber(137.3, 1134);
tempcow->BWT =GetStochasticNumber(10.13, 82.7);
tempcow->ACBWT =82.7;
tempcow->PAGD=GetStochasticNumber(.154, 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.495, 1062);
tempcow->PGA aviation=287.1;
tempcow->PGSD=6.06;
tempcow->AGE = MCOAGE;
tempcow->AGED = 0;
if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)) > (tempcow->PMWT) ){
tempcow->CDWT=tempcow->PMWT;
}
else if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)) <= (tempcow->PMWT) ){
tempcow->CDWT = ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE));
}
tempcow->SEAOP=0;
tempcow->SEAOP1=0;
tempcow->SEAOP2=0;
tempcow->BCS=5;
tempcow->COLR=COLR;
tempcow->DIE=0;
tempcow->REPLA=1;
tempcow->SOUR=1;
tempcow->CULL=0;
tempcow->CADAT=0;
tempcow->CNDAT=0;
tempcow->DETEC=0;
tempcow->PESTAMP=0;
tempcow->PGEST=0;
tempcow->IMPLANT=0;
tempcow->CUCYC=0;
tempcow->PREG=0;
tempcow->PARIE = 0;
tempcow->DPREG=0;
tempcow->DYSTO=0;
tempcow->FAII=0;
tempcow->FWEAN=0;
tempcow->PUBER=0;
tempcow->INELIG=1;
tempcow->Puberty=0;
tempcow->UNPREG=0;
tempcow->PPI=0;
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->FIDN=0;
tempcow->CSEX=0;
tempcow->GUWTK=0;
tempcow->CBWT=0;
tempcow->SERVI=0;
tempcow->PAST=1;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->MIDN=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->DAMBREED=tempcow->DAMBREED;
tempcow->GDAMBREED=0;
tempcow->GGDAMBREED=0;
tempcow->GSIREBREED=0;
tempcow->GGSIREBREED=0;
tempcow->GGGDAMBREED=0;
tempcow->GSIREBREED=0;
tempcow->GGSIREBREED=0;
tempcow->GGGSI'REBREED=0;
tempcow->CDWTK=(tempcow->CDWT*.453599);
tempcow->SBWTK=((tempcow->CDWT*.453599)* .96);
tempcow->EBWTK=tempcow->SBWTK * .891;
tempcow->EQSBW=((tempcow->SBWTK) * (478/(tempcow->PMWT*.453599)));
tempcow->EQEBW=tempcow->EQSBW * .891;
tempcow->SellCalf=0;
tempcow->SI'REBREED=0;
tempcow->BIRTHDATE=0;
tempcow->CALFSEAS=0;
tempcow->WEAN=0;
tempcow->WEANDATE=0;
tempcow->WEANWT=0;
tempcow->DHE'BTWT=0;
tempcow->DHE'TA DG=0;
tempcow->DHE(TMWT=0;
tempcow->MHE'TADG=0;
tempcow->MHE'TBWT=0;
tempcow->YFAIL=0;
tempcow->next=NULL;
AddAnimal(tempcow);
POHEF++;
REPLR--; 
BreedingCows++;
delete tempcow;
}
else if ( (BUYP == 0) && (TSTOP == 0) && (BUYRC==1) &&
(SelectW I Herd==0) && (REPLR>0)){
while (REPLR> 0){
AnimalRecord *tempcow = new AnimalRecord;
tempcow->ID=0;
tempcow->SEX=3;
if (YEAR<1){
tempcow->DAMBREED=HBREED;
tempcow->BREED=HBREED;
tempcow->PURE=PURE;
tempcow->SI'REBREED=HBREED;
tempcow->Fgeneration=0;
}
else if ((YEAR>=1) && (BuyPure == 0)){
tempcow->DAMBREED=BuyBreed + 1;
tempcow->BREED=BuyBreed + 1;
tempcow->PURE=PURE;
tempcow->SIREBREED=BuyBreed + 1;
tempcow->Fgeneration=0;
}  
else if ((YEAR>=1) && (BuyPure == 1)){
tempcow->PURE=PURE + 1;
tempcow->DAMBREED=BuyBreed + 1;
tempcow->BREED=BuyBreed + 1;
tempcow->SIREBREED=0;
tempcow->Fgeneration=0;
}
else if (tempcow->DAMBREED==1){
tempcow->PMWT=GetStochasticNumber(139.96, 1156);
tempcow->BWT =GetStochasticNumber(9.91, 80.9);
tempcow->ACBWT =80.9;
tempcow->PADG=GetStochasticNumber(.16, 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.0, 969);
tempcow->PGA =285.4;
tempcow->PGSD=6.02;
}
else if (tempcow->DAMBREED==2){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.56, 86.2);
tempcow->ACBWT =86.2;
tempcow->PADG=GetStochasticNumber(.145, 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85, 1038);
tempcow->PGA =288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==3){

tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.56, 86.2);
tempcow->ACBWT =86.2;
tempcow->PADG=GetStochasticNumber(.145, 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85, 1038);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
} else if (tempcow->DAMBREED==4){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.51, 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.154, 1.8);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.51, 988);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
} else if (tempcow->DAMBREED==5){
tempcow->PMWT=GetStochasticNumber(168.42, 1391);
tempcow->BWT =GetStochasticNumber(11.1, 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.158, 1.85);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218 , 1238);
tempcow->PGAVG=285.9;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==6){
tempcow->PMWT=GetStochasticNumber(189.24, 1563);
tempcow->BWT =GetStochasticNumber(10.94 , 89.3);
tempcow->ACBWT =89.3;
tempcow->PADG=GetStochasticNumber(.157 , 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(36.223 , 1350);
tempcow->PGAVG=287.5;
tempcow->PGSD=6.07;
}
else if (tempcow->DAMBREED==7){
tempcow->PMWT=GetStochasticNumber(156.67, 1294);
tempcow->BWT =GetStochasticNumber(10.54 , 86.0);
tempcow->ACBWT =86.0;
tempcow->PADG=GetStochasticNumber(.16 , 1.87);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(31.527 , 1175);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==8){
tempcow->PMWT=GetStochasticNumber(146.62, 1211);
tempcow->BWT =GetStochasticNumber(10.51 , 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);

antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(29.864 , 1113);
tempcow->PGAVG=288.1;
tempcow->PGSD=6.08;
}
else if (tempcow->DAMBREED==9){
tempcow->PMWT=GetStochasticNumber(184.16, 1521);
tempcow->BWT =GetStochasticNumber(11.1 , 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.154 , 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218 , 1238);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
}
else if (tempcow->DAMBREED==10){
tempcow->PMWT=GetStochasticNumber(149.17, 1232);
tempcow->BWT =GetStochasticNumber(10.58 , 86.4);
tempcow->ACBWT =86.4;
tempcow->PADG=GetStochasticNumber(.151 , 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844 , 1075);
tempcow->PGAVG=286.0;
tempcow->PGSD=6.03;
}
else if (tempcow->DAMBREED==11){
tempcow->PMWT=GetStochasticNumber(156.91, 1296);
tempcow->BWT = GetStochasticNumber(10.89, 88.9);
tempcow->ACBWT = 88.9;
tempcow->PADG = GetStochasticNumber(.157, 1.83);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(32.547, 1213);
tempcow->PGAVG = 286.3;
tempcow->PGSD = 6.04;
}
else if (tempcow->DAMBREED == 12){
tempcow->PMWT = GetStochasticNumber(137.3, 1134);
tempcow->BWT = GetStochasticNumber(10.13, 82.7);
tempcow->ACBWT = 82.7;
tempcow->PADG = GetStochasticNumber(.154, 1.81);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(28.495, 1062);
tempcow->PGAVG = 287.1;
tempcow->PGSD = 6.06;
}
tempcow->AGE = MCOAGE;
tempcow->AGED = 0;
if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)) > (tempcow->PMWT) ){
tempcow->CDWT = tempcow->PMWT;
}
else if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)) <= (tempcow->PMWT) ){
tempcow->CDWT = ((tempcow->BWT) + (tempcow->PADG * tempcow->AGE)));
}
tempcow->SEAOP = 0;
tempcow->SEAOP1 = 0;
tempcow->SEAOP2 = 0;
tempcow->BCS = 5;
tempcow->COLR=COLR;
tempcow->DIE=0;
tempcow->REPLA=1;
tempcow->SOUR=1;
tempcow->CULL=0;
tempcow->CADAT=0;
tempcow->CNDAT=0;
tempcow->DETEC=0;
tempcow->PGEST=0;
tempcow->IMPLANT=0;
tempcow->CUCYC=0;
tempcow->PREG=0;
tempcow->PARTIE = 0;
tempcow->DREG=0;
tempcow->DYSTO=0;
tempcow->FAIL=0;
tempcow->FWEAN=0;
tempcow->PUBER=0;
tempcow->INELIG=1;
tempcow->Puberty=0;
tempcow->UNPREG=0;
tempcow->PPI=0;
tempcow->LACT=0;
tempcow->LDAY=0;
tempcow->FIDN=0;
tempcow->CSEX=0;
tempcow->GUWTK=0;
tempcow->CBWT=0;
tempcow->SERVI=0;
tempcow->PAST=1;
tempcow->CIDN=0;
tempcow->CBIRTHDATE=0;
tempcow->MIDN=0;
tempcow->MCMFC=0;
tempcow->MCSNF=0;
tempcow->YN=0;
tempcow->YEN=0;
tempcow->DAMBRD=tempcow->DAMBRD;
tempcow->GDAMBRD=0;
tempcow->GGDAMBRD=0;
tempcow->GGGDAMBRD=0;
tempcow->GSIREBREED=0;
tempcow->GGSIREBREED=0;
tempcow->GGGSIREBREED=0;
tempcow->CDWTK=(tempcow->CDWT*.453599);
tempcow->SBWTK=((tempcow->CDWT*.453599)* .96);
tempcow->EBWTK=tempcow->SBWTK *.891;
tempcow->EQSBW=((tempcow->SBWTK) * (478/(tempcow->PMWT*.453599)));
tempcow->EQEBW=tempcow->EQSBW * .891;
tempcow->SellCalf=0;
tempcow->SIREBREED=0;
tempcow->BIRTHDATE=0;
tempcow->CALFSEAS=0;
tempcow->WEAN=0;
tempcow->WEANDATE=0;
tempcow->WEANWT=0;
tempcow->DHETBWT=0;
tempcow->DHETADG=0;
tempcow->DHETMWT=0;
tempcow->MHETADG=0;
tempcow->MHETBWT=0;
tempcow->YFAIL=0;
tempcow->next=NULL;
AddAnimal(tempcow);
POHEF++;
REPLR--;
BreedingCows++;
delete tempcow;
}
}
}
if ( (TEST==1) && (REPLR>0) ){
    if ( (REPLR>0) && (NumAvailRepla>0) && (SelectWIHerd==1) ){
        AnimalRecord *animalPointer;
        int StartPosition;
        animalPointer = GetHeadAnimal();
        animalPointer = animalPointer->next;
        StartPosition = animalPointer->ID;
        while ( (REPLR>0) && (NumAvailRepla>0) ){
            animalPointer=GetSex(0, StartPosition);
            while  ( (animalPointer != NULL) && (REPLR>0) && (NumAvailRepla>0) ){
                if ( (animalPointer->REPLA == 0) && (animalPointer->AGE>0) ){
                    REPLR--;
                    NumAvailRepla--;
                    animalPointer->REPLA = 1;
                    SELEC++;
                    BreedingReplacement++;
                }
                animalPointer = animalPointer->next;
                if (animalPointer != NULL){
                    StartPosition = animalPointer->ID;
                }
                animalPointer = GetSex(0, StartPosition);
            }
        }
    }else if ( (BUYRC==1) && ( ((NumAvailRepla==0) && (SelectWIHerd==1)) || (SelectWIHerd==0) )&& (REPLR>0)){
while (REPLR>0) {
    AnimalRecord *tempcow = new AnimalRecord;
    tempcow->ID=0;
    tempcow->SEX=3;
    if (YEAR<1) {
        tempcow->DAMBREED=HBREED;
        tempcow->PURE=PURE;
        tempcow->SIREBREED=HBREED;
        tempcow->Fgeneration=0;
    }
    else if ((YEAR>=1) && (BuyPure == 0)) {
        tempcow->DAMBREED=BuyBreed + 1;
        tempcow->BREED=BuyBreed + 1;
        tempcow->PURE=PURE;
        tempcow->SIREBREED=BuyBreed + 1;
        tempcow->Fgeneration=0;
    }
    else if ((YEAR>=1) && (BuyPure == 1)) {
        tempcow->PURE=PURE + 1;
        tempcow->DAMBREED=BuyBreed + 1;
        tempcow->BREED=BuyBreed + 1;
        tempcow->SIREBREED=0;
        tempcow->Fgeneration=0;
    }
    if (tempcow->DAMBREED==1) {
        tempcow->PMWT=GetStochasticNumber(139.96, 1156);
        tempcow->BWT =GetStochasticNumber(9.91, 80.9);
        tempcow->ACBWT =80.9;
        tempcow->PADG=GetStochasticNumber(.16, 1.87);
        tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
        tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
        tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
        tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
        tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
        tempcow->PMSWT=GetStochasticNumber(26.0, 969);
        tempcow->PGAVG=285.4;
        tempcow->PGSD=6.02;
    }
    else if (tempcow->DAMBREED==2) {
        tempcow->PMWT=GetStochasticNumber(143.84, 1188);
        tempcow->BWT =GetStochasticNumber(10.56, 86.2);
        tempcow->ACBWT =86.2;
        tempcow->PADG=GetStochasticNumber(.145, 1.70);
        tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
        tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==3){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.56 , 86.2);
tempcow->ACBWT =86.2;
tempcow->PADG=GetStochasticNumber(.145 , 1.70);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(27.85 , 1038);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==4){
tempcow->PMWT=GetStochasticNumber(143.84, 1188);
tempcow->BWT =GetStochasticNumber(10.51 , 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.154 , 1.8 );
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(26.51 , 988);
tempcow->PGAVG=288.6;
tempcow->PGSD=6.09;
}
else if (tempcow->DAMBREED==5){
tempcow->PMWT=GetStochasticNumber(168.42, 1391);
tempcow->BWT = GetStochasticNumber(11.1, 90.6);
tempcow->ACBWT = 90.6;
tempcow->PADG = GetStochasticNumber(.158, 1.85);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(33.218, 1238);
tempcow->PGAVG = 285.9;
tempcow->PGSD = 6.03;
}
else if (tempcow->DAMBREED == 6){
tempcow->PMWT = GetStochasticNumber(189.24, 1563);
tempcow->BWT = GetStochasticNumber(10.94, 89.3);
tempcow->ACBWT = 89.3;
tempcow->PADG = GetStochasticNumber(.157, 1.83);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT = GetStochasticNumber(36.223, 1350);
tempcow->PGAVG = 287.5;
tempcow->PGSD = 6.07;
}
else if (tempcow->DAMBREED == 7){
tempcow->PMWT = GetStochasticNumber(156.67, 1294);
tempcow->BWT = GetStochasticNumber(10.54, 86.0);
tempcow->ACBWT = 86.0;
tempcow->PADG = GetStochasticNumber(.16, 1.87);
tempcow->UADG = GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1 = GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2 = GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3 = GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4 = GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(31.527, 1175);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==8){
tempcow->PMWT=GetStochasticNumber(146.62, 1211);
tempcow->BWT =GetStochasticNumber(10.51, 85.8);
tempcow->ACBWT =85.8;
tempcow->PADG=GetStochasticNumber(.151, 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(29.864, 1113);
tempcow->PGAVG=288.1;
tempcow->PGSD=6.08;
}
else if (tempcow->DAMBREED==9){
tempcow->PMWT=GetStochasticNumber(184.16, 1521);
tempcow->BWT =GetStochasticNumber(11.1, 90.6);
tempcow->ACBWT =90.6;
tempcow->PADG=GetStochasticNumber(.154, 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685), antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685), antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685), antiADG4);
tempcow->PMSWT=GetStochasticNumber(33.218, 1238);
tempcow->PGAVG=285.4;
tempcow->PGSD=6.02;
}
else if (tempcow->DAMBREED==10){
tempcow->PMWT=GetStochasticNumber(149.17, 1232);
tempcow->BWT =GetStochasticNumber(10.58, 86.4);
tempcow->ACBWT =86.4;
tempcow->PADG=GetStochasticNumber(.151, 1.76);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), antiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685), antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.844, 1075);
tempcow->PGAVG=286.0;
tempcow->PGSD=6.03;
}  
else if (tempcow->DAMBREED==11){
tempcow->PMWT=GetStochasticNumber(156.91, 1296);
tempcow->BWT =GetStochasticNumber(10.89, 88.9);
tempcow->ACBWT =88.9;
tempcow->PADG=GetStochasticNumber(.157, 1.83);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(32.547, 1213);
tempcow->PGAVG=286.3;
tempcow->PGSD=6.04;
}
else if (tempcow->DAMBREED==12){
tempcow->PMWT=GetStochasticNumber(137.3, 1134);
tempcow->BWT =GetStochasticNumber(10.13, 82.7);
tempcow->ACBWT =82.7;
tempcow->PADG=GetStochasticNumber(.154, 1.81);
tempcow->UADG=GetStochasticNumber((antiADG/11.685), an-
tiADG);
tempcow->UADG1=GetStochasticNumber((antiADG1/11.685),
antiADG1);
    tempcow->UADG2=GetStochasticNumber((antiADG2/11.685),
antiADG2);
tempcow->UADG3=GetStochasticNumber((antiADG3/11.685),
antiADG3);
tempcow->UADG4=GetStochasticNumber((antiADG4/11.685),
antiADG4);
tempcow->PMSWT=GetStochasticNumber(28.495, 1062);
tempcow->PGAVG=287.1;
tempcow->PGSD=6.06;
}  
tempcow->AGE = MCOAGE;
tempcow->AGED = 0;
if ( ((tempcow->BWT) + (tempcow->PADG * tempcow->
>AGE)) > (tempcow->PMWT)) {
    tempcow->CDWT = tempcow->PMWT;
} else if ( ((tempcow->BWT) + (tempcow->PADG * tempcow-
>AGE)) <= (tempcow->PMWT)) {
    tempcow->CDWT = ((tempcow->BWT) + (tempcow->PADG * tempcow-
>AGE));
} tempcow->SEAOP = 0;
    tempcow->SEAOP1 = 0;
    tempcow->SEAOP2 = 0;
    tempcow->BCS = 5;
    tempcow->COLR = COLR;
    tempcow->DIE = 0;
    tempcow->REPLA = 1;
    tempcow->SOUR = 1;
    tempcow->CULL = 0;
    tempcow->CADAT = 0;
    tempcow->CNDAT = 0;
    tempcow->DETEC = 0;
    tempcow->PGEST = 0;
    tempcow->IMPLANT = 0;
    tempcow->CUCYC = 0;
    tempcow->PREG = 0;
    tempcow->PARIE = 0;
    tempcow->DPREG = 0;
    tempcow->DYSTO = 0;
    tempcow->FAIL = 0;
    tempcow->FWEAN = 0;
    tempcow->PUBER = 0;
    tempcow->INELIG = 1;
    tempcow->Puberty = 0;
    tempcow->UNPREG = 0;
    tempcow->PPI = 0;
    tempcow->LACT = 0;
    tempcow->LDAY = 0;
    tempcow->FIDN = 0;
    tempcow->CSEX = 0;
    tempcow->GUWTK = 0;
    tempcow->CBWT = 0;
    tempcow->SERVI = 0;
    tempcow->PAST = 1;
    tempcow->CIDN = 0;
    tempcow->CBIRTHDATE = 0;
    tempcow->MIDN = 0;
    tempcow->MCMFC = 0;
    tempcow->MCSNF = 0;
    tempcow->YN = 0;
    tempcow->YEN = 0;
    tempcow->DAMBREED = tempcow->DAMBREED;
    tempcow->GDAMBREED = 0;
tempcow->GGDAMBREED=0;
tempcow->GGGDAMBREED=0;
tempcow->GSIREBREED=0;
tempcow->GGSIREBREED=0;
tempcow->GGGSIREBREED=0;
tempcow->CDWTK=(tempcow->CDWT*.453599);
tempcow->SBWTK=((tempcow->CDWT*.453599)* .96);
tempcow->EBWTK=tempcow->SBWTK * .891;
tempcow->EQSBW=((tempcow->SBWTK) * (478/(tempcow->PMWT*.453599)));
tempcow->EQEBW=tempcow->EQSBW * .891;
tempcow->SellCalf=0;
tempcow->SIREBREED=0;
tempcow->BIRTHDATE=0;
tempcow->CALFSEAS=0;
tempcow->WEAN=0;
tempcow->WEANDATE=0;
tempcow->WEANWT=0;
tempcow->DHETBWT=0;
tempcow->DHETADG=0;
tempcow->DHETMWT=0;
tempcow->MHETADG=0;
tempcow->MHETBWT=0;
tempcow->YFAIL=0;
tempcow->next=NULL;
AddAnimal(tempcow);
POHEF++;
REPLR--;
BreedingCows++;
delete tempcow;
}
}

int Cow::GetOriginalNumOfCows()
{
    return COWS;
}

int Cow::DeadCattle(AnimalRecord * animal){
    if ( (animal->DIE==1) && (animal->AGE==animal->AGED) ){
        if ( (animal->SEX==3) && (animal->CULL==0) ){
            BreedingCows--;
        } else if( (animal->SEX==3) && (animal->CULL==1) ){
            BreedingCows--;
            CulledCows--;
        } else if ( (animal->SEX ==4) && (animal->REPLA==1) ){
            BreedingCows--;
        }
    }
}

633
else if ( (animal->SEX ==0) && (animal->REPLA==1) ){
    BreedingReplacement--;
}
}
else if ( (animal->DIE==1) && (animal->AGE>animal->AGED) ){
    DeleteAnimal(animal->ID);
    return 1;
}
return 0;
}

void Cow::ResetCowConCntr()
{
    AgedCulls=0;
    PregCullCow=0;
    PNUM=0;
    NumAvailRepla=0;
    NHFC1=0;
    NSTC1=0;
    TWHS1=0;
    TWSS1=0;
    ACDAT=0;
    AvServConcp=0;
    AWNWT=0;
    CLNUM=0;
    CONCE=0;
    CONCE1=0;
    BreedingCows=COWS;
    BreedingReplacement=0;
    CulledCows=0;
    CNRAT=0;
    CNUMB=0;
    CNUMBH=0;
    CNUMBS=0;
    CyclHeifer1=0;
    CyclHeifer2=0;
    CyclHeifer3=0;
    CyclHeifer21=0;
    CyclHeifer22=0;
    CyclHeifer23=0;
    PregHef1=0;
    PregHef2=0;
    PregHef3=0;
    PregHef21=0;
    PregHef22=0;
    PregHef23=0;
    CyclCow1=0;
    CyclCow2=0;
    CyclCow3=0;
CowsInElig=0;
CowsInElig1=0;
CowsElig1=0;
CalfCows=0;
CalfCows1=0;
WNWT=0;
TFWEANAGE=0;
TSWEANAGE=0;
AVFWEANAGE=0;
AVSWEANAGE=0;
FCALF=0;
SCALF=0;
BCScore[0] = 0;
BCScore[1] = 0;
BCScore[2] = 0;
BCScore[3] = 0;
BCScore[4] = 0;
BCScore[5] = 0;
BCScore[6] = 0;
BCScore[7] = 0;
BCScore[8] = 0;
BCScore[9] = 0;
AgeBreeders[0]=0;
AgeBreeders[1]=0;
AgeBreeders[2]=0;
AgeBreeders[3]=0;
AgeBreeders[4]=0;
AgeBreeders[5]=0;
AgeBreeders[6]=0;
AgeBreeders[7]=0;
AgeBreeders[8]=0;
AgeBreeders[9]=0;
AgeBreeders[10]=0;
AgeBreeders[11]=0;
AgeBreeders[12]=0;
AgeBreeders[13]=0;
AgeBreeders[14]=0;
NumCowsYend=0;
NumStkrYEnd=0;
TOTALAGE=0;
}

Forage.cpp
#include "stdafx.h"
#include "HerdSim.h"
#include "HerdSimDlg.h"
#include <iostream.h>
#include "forage.h"
#include "nutrit.h"
#include "tools.h"

int FARMSIZE;
int TERR;
extern double PRAC;
extern double PRAS;
extern double RESPS;
extern double RESP;
extern int currentDate;
extern int YEAR;
int CurrentSIZE;
int CurrentSIZES;
int SREG;
double Tc;
double Tp;
double FANEM;
double FAME;
double FANEG;
double TDNP;
double PAVAIL;
double CarryOver;
double TGRO;
double TTGRO;
double hayAVAIL;
double mainFDM;
double SELLHAY;
double HAYCOST;
extern int currentDate;
int dateOUT[7];
int dateIN[7];
int permPAST;
int hayLANDS;
int woodLANDS;
int dateROUT[7];
int dateRIN[7];
int permRPAST;
int hayRLANDS;
int woodRLANDS;
double DaySen;
double TotAnnSen;

Forage::Forage(){
    TERR = 2;
    PRAC = 20.00;
    RESP = 100.00;
    PRAS = 20.00;
RESPS = 100.00;
SLOE=0;
SREG=0;
FR[0]=0;
FR[1]=0;
FR[2]=0.45;
FR[3]=0;
FR[4]=.15;
FR[5]=0;
FR[6]=0.4;
FR[7]=0;
FR[8]=0;
FR[9]=0;
FR[10]=0;
FR[12]=0;
FR[13]=0;
FR[14]=0;
FR[15]=0;
FR[16]=0;
FR[17]=0;
FR[18]=0;
FR[19]=0;
FR[20]=0;
FR[21]=0;
FR[22]=0;
Tc=0;
Tp=0;
FAME=0;
FANEM=0;
FANEG=0;
TDNP=0;
TAPG=0;
PAVAIL=0;
CarryOver=0;
DaySen=0;
TotAnnSen=0;
FARMSIZE;
APRILGROWTH=0;
AUGUSTGROWTH=0;
DECEMBERGROWTH=0;
FEBRUARYGROWTH=0;
JANUARYGROWTH=0;
JULYGROWTH=0;
JUNEGROWTH=0;
MARCHGROWTH=0;
MAYGROWTH=0;
NOVEMBERGROWTH=0;
OCTOBERGROWTH=0;
SEPTEMBERGROWTH=0;
RAINAPRIL=0;
RAIN AUGUST = 0;
RAIN DECEMBER = 0;
RAIN FEBRUARY = 0;
RAIN JANUARY = 0;
RAIN JULY = 0;
RAIN JUNE = 0;
RAIN MARCH = 0;
RAIN MAY = 0;
RAIN NOVEMBER = 0;
RAIN OCTOBER = 0;
RAIN SEPTEMBER = 0;
TGRO = 0;
Current SIZE = 0;
Current SIZES = 0;
SLOE = 10;
EROS = 2;
FERT = 1;
GRAZ = 1;
WEDS = 3;
TERR = 2;
SREG = 3;
main FDM = 0;
SELL HAY = 60.00;
HAY COST = 31.04;
pasture = 1;
dateOut Month[1] = 3;
dateOut Month[2] = 3;
dateOut Month[3] = 3;
dateOut Month[4] = 3;
dateOut Month[5] = 3;
dateOut Month[6] = 3;
dateOut Day[1] = 1;
dateOut Day[2] = 1;
dateOut Day[3] = 1;
dateOut Day[4] = 1;
dateOut Day[5] = 1;
dateOut Day[6] = 1;
date OUT[1] = 60;
date OUT[2] = 60;
date OUT[3] = 60;
date OUT[4] = 60;
date OUT[5] = 60;
date OUT[6] = 60;
date In Month[1] = 7;
date In Month[2] = 7;
date In Month[3] = 7;
date In Month[4] = 7;
date In Month[5] = 7;
date In Month[6] = 7;
date In Day[1] = 1;
date In Day[2] = 1;
dateInDay[3]=1;
dateInDay[4]=1;
dateInDay[5]=1;
dateInDay[6]=1;
dateIN[1]=182;
dateIN[2]=182;
dateIN[3]=182;
dateIN[4]=182;
dateIN[5]=182;
dateIN[6]=182;
permPAST=0;
TPAS[1]=220;
TPAS[2]=0;
TPAS[3]=0;
TPAS[4]=0;
TPAS[5]=0;
TPAS[6]=0;
HLAND[1]=75;
HLAND[2]=0;
HLAND[3]=0;
HLAND[4]=0;
HLAND[5]=0;
HLAND[6]=0;
WLAND[1]=5;
WLAND[2]=0;
WLAND[3]=0;
WLAND[4]=0;
WLAND[5]=0;
WLAND[6]=0;
hayLANDS=0;
woodLANDS=0;
rpasture=0;
dateROutMonth[1]=3;
dateROutMonth[2]=3;
dateROutMonth[3]=3;
dateROutMonth[4]=3;
dateROutMonth[5]=3;
dateROutMonth[6]=3;
dateROUT[1]=60;
dateROUT[2]=60;
dateROUT[3]=60;
dateROUT[4]=60;
dateROUT[5]=60;
dateROUT[6]=60;
dateRInMonth[1]=7;
dateRInMonth[2]=7;
dateRInMonth[3]=7;
dateRInMonth[4]=7;
dateRInMonth[5]=7;
dateRInMonth[6]=7;
dateRInDay[1]=1;
dateRInDay[2]=1;
dateRInDay[3]=1;
dateRInDay[4]=1;
dateRInDay[5]=1;
dateRInDay[6]=1;
dateRIN[1]=182;
dateRIN[2]=182;
dateRIN[3]=182;
dateRIN[4]=182;
dateRIN[5]=182;
dateRIN[6]=182;
permRPAST=0;
TRPAS[1]=0;
TRPAS[2]=0;
TRPAS[3]=0;
TRPAS[4]=0;
TRPAS[5]=0;
TRPAS[6]=0;
HRLAND[1]=0;
HRLAND[2]=0;
HRLAND[3]=0;
HRLAND[4]=0;
HRLAND[5]=0;
HRLAND[6]=0;
WRLAND[1]=0;
WRLAND[2]=0;
WRLAND[3]=0;
WRLAND[4]=0;
WRLAND[5]=0;
WRLAND[6]=0;
hayRLANDS=0;
woodRLANDS=0;
}

void Forage::UseInput5(){
  CInput5* input=new CInput5;
  input->m_erosion=EROS-1;
  input->m_graze=GRAZ-1;
  input->m_maint=FERT;
  input->m_region=SREG-1;
  input->m_rentalrate=PRAC;
  input->m_respadd=RESPS;
  input->m_respmaint=RESP;
  input->m_sellhay=SELLHAY;
  input->m_haycost=HAYCOST;
  input->m_erosion=EROS-1;
  input->m_graze=GRAZ-1;
  input->m_maint=FERT;
  input->m_region=SREG-1;
  input->m_rentalrate=PRAC;
  input->m_respadd=RESPS;
  input->m_respmaint=RESP;
  input->m_sellhay=SELLHAY;
  input->m_haycost=HAYCOST;
input->m_slope=SLOE;
input->m_terrain=TERR-1;
input->m_weed=WEDES-1;
int response=input->DoModal();
EROS=input->m_erosion+1;
GRAZ=input->m_graze+1;
FERT=input->m_maint;
SREG=input->m_region+1;
PRAC=input->m_rentalrate;
RESPS=input->m_respadd;
RESP=input->m_respmaint;
SELLHAY=input->m_sellhay;
HAYCOST=input->m_haycost;
SLOE=input->m_slope;
TERR=input->m_terrain+1;
WEDES=input->m_weed+1;
delete input;
if (response==IDC_NEXT){
    UseInput9();
}
return;

void Forage::UseInput6(){
    int response;
do{
    CInput6* input=new CInput6;
    input->m_alfalfa=FR[8];
    input->m_bahia=FR[13];
    input->m_bermuda100=FR[9];
    input->m_bermuda200=FR[10];
    input->m_bermuda250=FR[11];
    input->m_bermudar=FR[12];
    input->m_dallis=FR[14];
    input->m_fescue=FR[1];
    input->m_fescuelad=FR[2];
    input->m_kentucky=FR[3];
    input->m_kentuckyw=FR[4];
    input->m_kudza=FR[15];
    input->m_lespedza=FR[16];
    input->m_orchard=FR[5];
    input->m_orchardl=FR[6];
    input->m_orchardr=FR[7];
    input->m_pearl=FR[20];
    input->m_rye=FR[17];
    input->m_sericea=FR[18];
input->m_sudan=FR[21];
input->m_switch=FR[19];
input->m_total=TFR;
response=input->DoModal();
if(response==IDCANCEL){
delete input;
return;
}
}
FR[8]=input->m_alfalfa;
FR[13]=input->m_bahia;
FR[9]=input->m_bermuda100;
FR[10]=input->m_bermuda200;
FR[12]=input->m_bermudar;
FR[14]=input->m_dallis;
FR[1]=input->m_fescue;
FR[2]=input->m_fescuelad;
FR[3]=input->m_kentucky;
FR[4]=input->m_kentuckyw;
FR[15]=input->m_kudza;
FR[16]=input->m_lespedza;
FR[5]=input->m_orchard;
FR[6]=input->m_orchardl;
FR[7]=input->m_orchardr;
FR[20]=input->m_pearl;
FR[17]=input->m_rye;
FR[18]=input->m_sericea;
FR[21]=input->m_sudan;
FR[19]=input->m_switch;
delete input;
if (TFR != 1)
AfxMessageBox("Totals must equal 1.00", MB_OK);
}while(TFR!=1);
if (response==IDC_PREV){
UseInput10();
}
return;
}

void Forage::UseInput9()
{
CINPUT9* input=new CINPUT9;
int i;
input->m_tpas1 = TPAS[1];
input->m_tpas2 = TPAS[2];
input->m_tpas3 = TPAS[3];
input->m_tpas4 = TPAS[4];
input->m_tpas5 = TPAS[5];
input->m_tpas6 = TPAS[6];
input->m_hland1 = HLAND[1];
input->m_hland2 = HLAND[2];
input->m_hland3 = HLAND[3];
input->m_hland4 = HLAND[4];
input->m_hland5 = HLAND[5];
input->m_hland6 = HLAND[6];
input->m_wland1 = WLAND[1];
input->m_wland2 = WLAND[2];
input->m_wland3 = WLAND[3];
input->m_wland4 = WLAND[4];
input->m_wland5 = WLAND[5];
input->m_wland6 = WLAND[6];
input->m_monthin1 = dateInMonth[1];
input->m_monthin2 = dateInMonth[2];
input->m_monthin3 = dateInMonth[3];
input->m_monthin4 = dateInMonth[4];
input->m_monthin5 = dateInMonth[5];
input->m_monthin6 = dateInMonth[6];
input->m_dayin1 = dateInDay[1];
input->m_dayin2 = dateInDay[2];
input->m_dayin3 = dateInDay[3];
input->m_dayin4 = dateInDay[4];
input->m_dayin5 = dateInDay[5];
input->m_dayin6 = dateInDay[6];
input->m_monthout1 = dateOutMonth[1];
input->m_monthout2 = dateOutMonth[2];
input->m_monthout3 = dateOutMonth[3];
input->m_monthout4 = dateOutMonth[4];
input->m_monthout5 = dateOutMonth[5];
input->m_monthout6 = dateOutMonth[6];
input->m_dayout1 = dateOutDay[1];
input->m_dayout2 = dateOutDay[2];
input->m_dayout3 = dateOutDay[3];
input->m_dayout4 = dateOutDay[4];
input->m_dayout5 = dateOutDay[5];
input->m_dayout6 = dateOutDay[6];
input->m_numpasture = pasture;
int response = input->DoModal();
TPAS[1] = input->m_tpas1;
TPAS[2] = input->m_tpas2;
TPAS[3] = input->m_tpas3;
TPAS[4] = input->m_tpas4;
TPAS[5] = input->m_tpas5;
TPAS[6] = input->m_tpas6;
HLAND[1] = input->m_hland1;
HLAND[2] = input->m_hland2;
HLAND[3] = input->m_hland3;
HLAND[4] = input->m_hland4;
HLAND[5] = input->m_hland5;
HLAND[6] = input->m_hland6;
WLAND[1] = input->m_wland1;
WLAND[2] = input->m_wland2;
WLAND[3] = input->m_wland3;
WLAND[4] = input->m_wland4;
WLAND[5] = input->m_wland5;
WLAND[6] = input->m_wland6;
dateInMonth[1] = input->m_monthin1;
dateInMonth[2] = input->m_monthin2;
dateInMonth[3] = input->m_monthin3;
dateInMonth[4] = input->m_monthin4;
dateInMonth[5] = input->m_monthin5;
dateInMonth[6] = input->m_monthin6;
dateInDay[1] = input->m_dayin1;
dateInDay[2] = input->m_dayin2;
dateInDay[3] = input->m_dayin3;
dateInDay[4] = input->m_dayin4;
dateInDay[5] = input->m_dayin5;
dateInDay[6] = input->m_dayin6;
dateOutMonth[1] = input->m_monthout1;
dateOutMonth[2] = input->m_monthout2;
dateOutMonth[3] = input->m_monthout3;
dateOutMonth[4] = input->m_monthout4;
dateOutMonth[5] = input->m_monthout5;
dateOutMonth[6] = input->m_monthout6;
dateOutDay[1] = input->m_dayout1;
dateOutDay[2] = input->m_dayout2;
dateOutDay[3] = input->m_dayout3;
dateOutDay[4] = input->m_dayout4;
dateOutDay[5] = input->m_dayout5;
dateOutDay[6] = input->m_dayout6;
dateOUT[1] = Julian(dateOutMonth[1], dateOutDay[1]);
dateOUT[2] = Julian(dateOutMonth[2], dateOutDay[2]);
dateOUT[3] = Julian(dateOutMonth[3], dateOutDay[3]);
dateOUT[4] = Julian(dateOutMonth[4], dateOutDay[4]);
dateOUT[5] = Julian(dateOutMonth[5], dateOutDay[5]);
dateOUT[6] = Julian(dateOutMonth[6], dateOutDay[6]);
dateIN[1] = Julian(dateInMonth[1], dateInDay[1]);
dateIN[2] = Julian(dateInMonth[2], dateInDay[2]);
dateIN[3] = Julian(dateInMonth[3], dateInDay[3]);
dateIN[4] = Julian(dateInMonth[4], dateInDay[4]);
dateIN[5] = Julian(dateInMonth[5], dateInDay[5]);
dateIN[6] = Julian(dateInMonth[6], dateInDay[6]);
pasture = input->m_numpasture;
delete input;
if (response==IDC_NEXT){
    UseInput10();
}
return;
void Forage::UseInput10()
{
    CInput10* input = new CInput10;
    int i;
    input->m_trpas1 = TRPAS[1];
    input->m_trpas2 = TRPAS[2];
    input->m_trpas3 = TRPAS[3];
    input->m_trpas4 = TRPAS[4];
    input->m_trpas5 = TRPAS[5];
    input->m_trpas6 = TRPAS[6];
    input->m_hrland1 = HRLAND[1];
    input->m_hrland2 = HRLAND[2];
    input->m_hrland3 = HRLAND[3];
    input->m_hrland4 = HRLAND[4];
    input->m_hrland5 = HRLAND[5];
    input->m_hrland6 = HRLAND[6];
    input->m_wrland1 = WRLAND[1];
    input->m_wrland2 = WRLAND[2];
    input->m_wrland3 = WRLAND[3];
    input->m_wrland4 = WRLAND[4];
    input->m_wrland5 = WRLAND[5];
    input->m_wrland6 = WRLAND[6];
    input->m_monthrin1 = dateRInMonth[1];
    input->m_monthrin2 = dateRInMonth[2];
    input->m_monthrin3 = dateRInMonth[3];
    input->m_monthrin4 = dateRInMonth[4];
    input->m_monthrin5 = dateRInMonth[5];
    input->m_monthrin6 = dateRInMonth[6];
    input->m_dayrin1 = dateRInDay[1];
    input->m_dayrin2 = dateRInDay[2];
    input->m_dayrin3 = dateRInDay[3];
    input->m_dayrin4 = dateRInDay[4];
    input->m_dayrin5 = dateRInDay[5];
    input->m_dayrin6 = dateRInDay[6];
    input->m_monthrout1 = dateROutMonth[1];
    input->m_monthrout2 = dateROutMonth[2];
    input->m_monthrout3 = dateROutMonth[3];
    input->m_monthrout4 = dateROutMonth[4];
    input->m_monthrout5 = dateROutMonth[5];
    input->m_monthrout6 = dateROutMonth[6];
    input->m_dayrout1 = dateROutDay[1];
    input->m_dayrout2 = dateROutDay[2];
    input->m_dayrout3 = dateROutDay[3];
    input->m_dayrout4 = dateROutDay[4];
    input->m_dayrout5 = dateROutDay[5];
    input->m_dayrout6 = dateROutDay[6];

    input->m_numrpasture = rpasture;
    int response = input->DoModal();
    TRPAS[1] = input->m_trpas1;
    TRPAS[2] = input->m_trpas2;
    TRPAS[3] = input->m_trpas3;
TRPAS[4] = input->m_trpas4;
TRPAS[5] = input->m_trpas5;
TRPAS[6] = input->m_trpas6;
HRLAND[1] = input->m_hrland1;
HRLAND[2] = input->m_hrland2;
HRLAND[3] = input->m_hrland3;
HRLAND[4] = input->m_hrland4;
HRLAND[5] = input->m_hrland5;
HRLAND[6] = input->m_hrland6;
WRLAND[1] = input->m_wrland1;
WRLAND[2] = input->m_wrland2;
WRLAND[3] = input->m_wrland3;
WRLAND[4] = input->m_wrland4;
WRLAND[5] = input->m_wrland5;
WRLAND[6] = input->m_wrland6;
dateRInMonth[1] = input->m_monthrin1;
dateRInMonth[2] = input->m_monthrin2;
dateRInMonth[3] = input->m_monthrin3;
dateRInMonth[4] = input->m_monthrin4;
dateRInMonth[5] = input->m_monthrin5;
dateRInMonth[6] = input->m_monthrin6;
    dateRInDay[1] = input->m_dayrin1;
dateRInDay[2] = input->m_dayrin2;
dateRInDay[3] = input->m_dayrin3;
dateRInDay[4] = input->m_dayrin4;
dateRInDay[5] = input->m_dayrin5;
dateRInDay[6] = input->m_dayrin6;
dateROutMonth[1] = input->m_monthrout1;
dateROutMonth[2] = input->m_monthrout2;
dateROutMonth[3] = input->m_monthrout3;
dateROutMonth[4] = input->m_monthrout4;
dateROutMonth[5] = input->m_monthrout5;
dateROutMonth[6] = input->m_monthrout6;
dateROutDay[1] = input->m_dayrout1;
dateROutDay[2] = input->m_dayrout2;
dateROutDay[3] = input->m_dayrout3;
dateROutDay[4] = input->m_dayrout4;
dateROutDay[5] = input->m_dayrout5;
dateROutDay[6] = input->m_dayrout6;
    dateROUT[1] = Julian(dateOutMonth[1], dateROutDay[1]);
dateROUT[2] = Julian(dateOutMonth[2], dateROutDay[2]);
dateROUT[3] = Julian(dateOutMonth[3], dateROutDay[3]);
dateROUT[4] = Julian(dateOutMonth[4], dateROutDay[4]);
dateROUT[5] = Julian(dateOutMonth[5], dateROutDay[5]);
dateROUT[6] = Julian(dateOutMonth[6], dateROutDay[6]);
dateRIN[1] = Julian(dateRInMonth[1], dateRInDay[1]);
dateRIN[2] = Julian(dateRInMonth[2], dateRInDay[2]);
dateRIN[3] = Julian(dateRInMonth[3], dateRInDay[3]);
dateRIN[4] = Julian(dateRInMonth[4], dateRInDay[4]);
dateRIN[5] = Julian(dateRInMonth[5], dateRInDay[5]);
dateRIN[6] = Julian(dateRInMonth[6], dateRInDay[6]);
rpasture = input->m_numrpasture;
delete input;
if (response==IDC_NEXT){
    UseInput6();
}
else if (response==IDC_PREV){
    UseInput9();
}
return;

void Forage::Init(){
    UseInput5();
}

int Forage::Temperature(){
    double RegTemp[4][2][13];
    RegTemp[1][0][1]=2.87;
    RegTemp[1][1][1]=0.759;
    RegTemp[1][0][2]=4.3;
    RegTemp[1][1][2]=0.631;
    RegTemp[1][0][3]=8.95;
    RegTemp[1][1][3]=0.667;
    RegTemp[1][0][4]=13.81;
    RegTemp[1][1][4]=0.588;
    RegTemp[1][0][5]=18.84;
    RegTemp[1][1][5]=0.426;
    RegTemp[1][0][6]=23.2;
    RegTemp[1][1][6]=0.388;
    RegTemp[1][0][7]=25.46;
    RegTemp[1][1][7]=0.369;
    RegTemp[1][0][8]=24.85;
    RegTemp[1][1][8]=0.411;
    RegTemp[1][0][9]=21.53;
    RegTemp[1][1][9]=0.617;
    RegTemp[1][0][10]=15.53;
    RegTemp[1][1][10]=0.796;
    RegTemp[1][0][11]=10.56;
    RegTemp[1][1][11]=0.738;
    RegTemp[1][0][12]=5.38;
    RegTemp[1][1][12]=0.843;
    RegTemp[2][0][1]=1.15;
    RegTemp[2][1][1]=1.06;
    RegTemp[2][0][2]=2.79;
    RegTemp[2][1][2]=1.171;
    RegTemp[2][0][3]=7.88;
    RegTemp[2][1][3]=1.024;
    RegTemp[2][0][4]=12.93;
    RegTemp[2][1][4]=0.949;
    RegTemp[2][0][5]=17.83;
    RegTemp[2][1][5]=0.733;
RegTemp[2][0][6]=22.22;
RegTemp[2][1][6]=0.699;
RegTemp[2][0][7]=24.39;
RegTemp[2][1][7]=0.699;
RegTemp[2][0][8]=23.68;
RegTemp[2][1][8]=0.693;
RegTemp[2][0][9]=20.02;
RegTemp[2][1][9]=0.74;
RegTemp[2][0][10]=13.78;
RegTemp[2][1][10]=0.769;
RegTemp[2][0][11]=8.72;
RegTemp[2][1][11]=0.886;
RegTemp[2][0][12]=3.31;
RegTemp[2][1][12]=1.058;
RegTemp[3][0][1]=-.23;
RegTemp[3][1][1]=1.459;
RegTemp[3][0][2]=1.72;
RegTemp[3][1][2]=1.533;
RegTemp[3][0][3]=6.83;
RegTemp[3][1][3]=1.54;
RegTemp[3][0][4]=11.66;
RegTemp[3][1][4]=1.62;
RegTemp[3][0][5]=16.27;
RegTemp[3][1][5]=1.597;
RegTemp[3][0][6]=20.5;
RegTemp[3][1][6]=1.541;
RegTemp[3][0][7]=22.5;
RegTemp[3][1][7]=1.596;
RegTemp[3][0][8]=21.99;
RegTemp[3][1][8]=1.516;
RegTemp[3][0][9]=18.43;
RegTemp[3][1][9]=1.576;
RegTemp[3][0][10]=12.35;
RegTemp[3][1][10]=1.37;
RegTemp[3][0][11]=7.25;
RegTemp[3][1][11]=1.224;
RegTemp[3][0][12]=2.23;
RegTemp[3][1][12]=1.281;
if ( (currentDATE >= 1) && (currentDATE < 32)) {
  Tc=GetStochasticNumber(RegTemp[SREG][1][1],
  RegTemp[SREG][0][1]);
  if (currentDATE==1){
    Tp=GetStochasticNumber(RegTemp[SREG][1][12],
    RegTemp[SREG][0][12]);
  }
}
else if ( (currentDATE >= 32) && (currentDATE < 60)) {
  Tc=GetStochasticNumber(RegTemp[SREG][1][2],
  RegTemp[SREG][0][2]);
  if (currentDATE==32){
    Tp=GetStochasticNumber(RegTemp[SREG][1][1],

RegTemp[SREG][0][1]);
}

else if ( (currentDATE >= 60) && (currentDATE < 91)){
Tc=GetStochasticNumber(RegTemp[SREG][1][3],
RegTemp[SREG][0][3]);
if (currentDATE==60){
Tp=GetStochasticNumber(RegTemp[SREG][1][2],
RegTemp[SREG][0][2]);
}

else if ( (currentDATE >= 91) && (currentDATE < 121)){
Tc=GetStochasticNumber(RegTemp[SREG][1][4],
RegTemp[SREG][0][4]);
if (currentDATE==91){
Tp=GetStochasticNumber(RegTemp[SREG][1][3],
RegTemp[SREG][0][3]);
}

else if ( (currentDATE >= 121) && (currentDATE < 152)){
Tc=GetStochasticNumber(RegTemp[SREG][1][5],
RegTemp[SREG][0][5]);
if (currentDATE==121){
Tp=GetStochasticNumber(RegTemp[SREG][1][4],
RegTemp[SREG][0][4]);
}

else if ( (currentDATE >= 152) && (currentDATE < 182)){
Tc=GetStochasticNumber(RegTemp[SREG][1][6],
RegTemp[SREG][0][6]);
if (currentDATE==152){
Tp=GetStochasticNumber(RegTemp[SREG][1][5],
RegTemp[SREG][0][5]);
}

else if ( (currentDATE >= 182) && (currentDATE < 213)){
Tc=GetStochasticNumber(RegTemp[SREG][1][7],
RegTemp[SREG][0][7]);
if (currentDATE==182){
Tp=GetStochasticNumber(RegTemp[SREG][1][6],
RegTemp[SREG][0][6]);
}

else if ( (currentDATE >= 213) && (currentDATE < 244)){
Tc=GetStochasticNumber(RegTemp[SREG][1][8],
RegTemp[SREG][0][8]);
if (currentDATE==213){
Tp=GetStochasticNumber(RegTemp[SREG][1][7],
RegTemp[SREG][0][7]);
}
else if ( (currentDATE >= 244) && (currentDATE < 274))
{ Tc=GetStochasticNumber(RegTemp[SREG][1][9], RegTemp[SREG][0][9]);
 if (currentDATE==244)
{ Tp=GetStochasticNumber(RegTemp[SREG][1][8], RegTemp[SREG][0][8]);
}
}
else if ( (currentDATE >= 274) && (currentDATE < 305))
{ Tc=GetStochasticNumber(RegTemp[SREG][1][10], RegTemp[SREG][0][10]);
 if (currentDATE==274)
{ Tp=GetStochasticNumber(RegTemp[SREG][1][9], RegTemp[SREG][0][9]);
}
}
else if ( (currentDATE >= 305) && (currentDATE < 335))
{ Tc=GetStochasticNumber(RegTemp[SREG][1][11], RegTemp[SREG][0][11]);
 if (currentDATE==305)
{ Tp=GetStochasticNumber(RegTemp[SREG][1][10], RegTemp[SREG][0][10]);
}
}
else if ( (currentDATE >= 335) && (currentDATE < 365))
{ Tc=GetStochasticNumber(RegTemp[SREG][1][12], RegTemp[SREG][0][12]);
 if (currentDATE==335)
{ Tp=GetStochasticNumber(RegTemp[SREG][1][11], RegTemp[SREG][0][11]);
}
}
return 0;
}

void Forage::MainForage()
{
double FAVAIL[22]={0,3948,2738,2217,2953,3755,4820,5009,3359,5049,7266,5960,3087,2579,1510,1999,3841,2586,3907,4299,3330};
double FME[22]={0,2.1,2.128,2.40,2.50,2.15,2.16,2.086,2.302,2.265,2.265,2.319,2.335,1.92,2.33,2.35,2.405,2.44,2.405,1.89,2.20,2.40};
double FNEG[22]={0,.68,.704,.86,.94,.715,.725,.661,.847,.74,.74,.78,8,.80,.40,.80,.82,.86,.89,.86,.36,.68,.86};
double FNEM[22]={0,1.24,1.244,1.48,1.56,1.29,1.298,1.23,1.432,1.38,1.38,1.42,1.432,1.14,1.426,1.44,1.48,1.51,1.48,1.12,1.33,1.48};
double

double hayFDM[22]={0,.88,.89,.90,.91,.90,.91,.92,.90,.91,.644,.65,.665,.67,.665,.52,.61,.63};

tf{[315x57]}

int PFOR1[13]={0,5,10,35,50,38,15,5,8,18,38,13,5};
int PFOR2[13]={0,4,6,22,54,43,17,9,9,17,22,7,4};
int PFOR3[13]={0,0,1,13,20,33,27,13,5,8,9,2,0};
int PFOR4[13]={0,0,1,13,20,33,27,13,5,8,9,2,0};
int PFOR5[13]={0,4,7,28,40,38,16,8,10,14,20,6,4};
int PFOR6[13]={0,2,3,24,50,40,20,12,12,14,16,3,2};
int PFOR7[13]={0,2,4,23,58,47,28,14,14,16,19,4,2};
int PFOR8[13]={0,0,0,27,40,53,40,27,40,27,8,2,0};
int PFOR9[13]={0,0,0,16,20,30,50,40,30,10,2,0};
int PFOR10[13]={0,0,0,0,24,30,45,75,60,45,15,3,0};
int PFOR11[13]={0,7,13,37,51,22,29,62,55,33,18,17,11};
int PFOR12[13]={0,4,6,35,38,54,65,65,58,31,19,14,8};
int PFOR13[13]={0,0,0,0,4,7,39,46,39,29,17,4,0};
int PFOR14[13]={0,0,0,0,12,30,45,38,23,3,0,0};
int PFOR15[13]={0,0,0,0,4,18,29,29,18,12,8,0,0};
int PFOR16[13]={0,0,0,0,4,14,41,35,21,2,0,0};
int PFOR17[13]={0,8,16,30,42,56,47,12,0,0,7,8,7};
int PFOR18[13]={0,0,0,0,8,23,45,38,23,15,0,0,0};
int PFOR19[13]={0,0,0,5,14,35,47,55,46,24,7,0,0};
int PFOR20[13]={0,0,0,0,18,50,75,63,38,8,0,0};
int PFOR21[13]={0,0,0,0,19,53,80,67,40,8,0,0};
double RAINAVERAGE2[13]={0,3.12,3.18,3.66,3.25,4.16,3.56,4.23,4.05,3.55,3.88,3.42,3.16};
double RAINAVERAGE3[13]={0,2.82,2.96,3.51,3.36,4.06,3.55,4.13,3.68,3.50,3.64,3.23,2.97};
double RAINDEVIATION1[13]={0,.383,.350,.278,.235,.389,.423,.598,.609,.476,.365,.228,.244};
double RAINDEVIATION3[13]={0,.505,.511,.557,.512,.415,.500,.487,.422,.522,.526,.549,.470};
const double SENE=.0035714285714;

int i;
double AGRO=0;
double HAGRO=0;
double DGRO=0;
double PADJ=0;
double HPADJ=0;
double PMAIN;
double PMAN;
double SLOP;
double SOIL;
double WEED;
double ANNRAIN=0;
double APREP;
double MCRAIN1[13];
double MCRAIN2[13];
double MCRAIN3[13];
double MCARAIN1;
double MCARAIN2;
double MCARAIN3;

    HLAND[6] + 
    WLAND[6] + 

if (EROS==1){
    SOIL=.833333333;
}
else if (EROS==2){
    SOIL=1.166666667;
}
else if (EROS==3){
    SOIL=1.50;
}
else if (EROS==4){
    SOIL = 1.666666667;
}

if (GRAZ==1){
    PMAN=.75;
}
else if (GRAZ==2){
    PMAN=.90;
}
else if (GRAZ==3){
    PMAN=1.05;
}
if ((SLOE/100)<=.10){
    SLOP=1;
}
else if ( ((SLOE/100)> .10) && ((SLOE/100)<=.60) ){
    SLOP=(1- (.01*SLOE));
}
else if ((SLOE/100) > .60){
SLOP=0;
}
if (WEDS==1){
  WEED=.30;
}
else if (WEDS==2){
  WEED=.70;
}
else if (WEDS==3){
  WEED=.90;
}
else if (WEDS==4){
  WEED=1;
}
if (FERT==0){
  PMAIN=.50;
}
else if (FERT==1){
  PMAIN=1;
}
else if (FERT==2){
  PMAIN=1.15;
}
if ((currentDATE>=1) && (currentDATE<32)){
  if (SREG==1){
    if (currentDATE==1){
      MCRAIN1[1]=GetStochasticNumber(RAINDEVIATION1[1],
                                      RAINAVERAGE1[1]);
      RAINJANUARY=MCRAIN1[1];
    }
    if (RAINJANUARY>4.7390){
      PREP=1.57;
    }
    else if ((RAINJANUARY>4.3560) && (RAINJANUARY<=4.7390)){
      PREP=1.38;
    }
    else if ((RAINJANUARY>3.9730) && (RAINJANUARY<=4.3560)){
      PREP=1.19;
    }
    else if ((RAINJANUARY>3.2070) && (RAINJANUARY<=3.9730)){
      PREP=1;
    }
    else if ((RAINJANUARY>2.8240) && (RAINJANUARY<=3.2070)){
      PREP=.81;
    }
    else if ((RAINJANUARY>2.4410) && (RAINJANUARY<=2.8240)){
      PREP=.62;
    }
    else if (RAINJANUARY<=2.4410){
      PREP=.43;
    }
  }
}  
else if (SREG==2)
}  
else if (SREG==3)
}  
else if (currentDATE == 1)
MCRAIN2[1]=GetStochasticNumber(RAINDEVIATION2[1],
RAINAVERAGE2[1]);
RAINJANUARY=MCRAIN2[1];
}  
if (RAINJANUARY>3.9420)
PREP=1.57;
else if ((RAINJANUARY>3.6680) && (RAINJANUARY<=3.9420))
PREP=1.38;
else if ((RAINJANUARY>3.3940) && (RAINJANUARY<=3.6680))
PREP=1.19;
else if ((RAINJANUARY>2.8460) && (RAINJANUARY<=3.3940))
PREP=1;
else if ((RAINJANUARY>2.5720) && (RAINJANUARY<=2.8460))
PREP=.81;
else if ((RAINJANUARY>2.2980) && (RAINJANUARY<=2.5720))
PREP=.62;
else if (RAINJANUARY<=2.2980)
PREP=.43;

else if (SREG==3)
}  
else if (currentDATE == 1)
MCRAIN3[1]=GetStochasticNumber(RAINDEVIATION3[1],
RAINAVERAGE3[1]);
RAINJANUARY=MCRAIN3[1];
}  
if (RAINJANUARY>4.8030)
PREP=1.57;
else if ((RAINJANUARY>4.1220) && (RAINJANUARY<=4.8030))
PREP=1.38;
else if ((RAINJANUARY>3.4410) && (RAINJANUARY<=4.1220))
PREP=1.19;
else if ((RAINJANUARY>2.0790) && (RAINJANUARY<=3.4410))
PREP=1;
else if ((RAINJANUARY>1.3980) && (RAINJANUARY<=2.0790))
PREP=.81;
else if ((RAINJANUARY>.7170) && (RAINJANUARY<=1.3980)){
PREP=.62;
}
else if (RAINJANUARY<=.7170){
PREP=.43;
}
}
}
else if ((currentDATE>=32) && (currentDATE<60)){
if (SREG==1){
if (currentDATE==32){
MCRAIN1[2]=GetStochasticNumber(RAINDEV1[2],
RAINAVG1[2]);
RAINFEBRUARY=MCRAIN1[2];
}
if (RAINFEBRUARY>4.43){
PREP=1.57;
}
else if ((RAINFEBRUARY>4.08) && (RAINFEBRUARY<=4.43)){
PREP=1.38;
}
else if ((RAINFEBRUARY>3.73) && (RAINFEBRUARY<=4.08)){
PREP=1.19;
}
else if ((RAINFEBRUARY>3.03) && (RAINFEBRUARY<=3.73)){
PREP=1;
}
else if ((RAINFEBRUARY>2.68) && (RAINFEBRUARY<=3.03)){
PREP=.81;
}
else if ((RAINFEBRUARY>2.33) && (RAINFEBRUARY<=2.68)){
PREP=.62;
}
else if (RAINFEBRUARY<=2.33){
PREP=.43;
}
}
else if (SREG==2){
if (currentDATE==32){
MCRAIN2[2]=GetStochasticNumber(RAINDEV2[2],
RAINAVG2[2]);
RAINFEBRUARY=MCRAIN2[2];
}
if (RAINFEBRUARY>4.158){
PREP=1.57;
}
else if ((RAINFEBRUARY>3.832) && (RAINFEBRUARY<=4.158)){
PREP=1.38;
}
else if ((RAINFEBRUARY>3.506) && (RAINFEBRUARY<=3.832)){
PREP=1.19;
}
else if ((RAINFEBRUARY>2.854) && (RAINFEBRUARY<=3.506)) { PREP=1.00; }
else if ((RAINFEBRUARY>2.528) && (RAINFEBRUARY<=2.854)) { PREP=.81; }
else if ((RAINFEBRUARY>2.202) && (RAINFEBRUARY<=2.528)) { PREP=.62; }
else if (RAINFEBRUARY<=2.202) { PREP=.43; }
else if (SREG==3) {
  if (currentDATE==32) {
    MCRAIN3[2]=GetStochasticNumber(RAINDEVIATION3[2],
    RAINAVERAGE3[2]);
    RAINFEBRUARY=MCRAIN3[2];
  }
  if (RAINFEBRUARY>4.993) { PREP=1.57; }
  else if ((RAINFEBRUARY>4.2920) && (RAINFEBRUARY<=4.993)) { PREP=1.38; }
  else if ((RAINFEBRUARY>3.5910) && (RAINFEBRUARY<=4.2920)) { PREP=1.19; }
  else if ((RAINFEBRUARY>2.1890) && (RAINFEBRUARY<=3.5910)) { PREP=1; }
  else if ((RAINFEBRUARY>1.4880) && (RAINFEBRUARY<=2.1890)) { PREP=.81; }
  else if ((RAINFEBRUARY>.7870) && (RAINFEBRUARY<=1.4880)) { PREP=.62; }
  else if (RAINFEBRUARY<=.7870) { PREP=.43; }
}
else if ((currentDATE>=60) && (currentDATE<91)) {
  if (SREG==1) {
    if (currentDATE==60) {
      MCRAIN1[3]=GetStochasticNumber(RAINDEVIATION1[3],
      RAINAVERAGE1[3]);
      RAINMARCH=MCRAIN1[3];
    }
    if (RAINMARCH>4.6940) { PREP=1.57; }
else if ((RAINMARCH>4.4160) && (RAINMARCH<=4.6940)){
    PREP=1.38;
}
else if ((RAINMARCH>4.1380) && (RAINMARCH<=4.4160)){
    PREP=1.19;
}
else if ((RAINMARCH>3.5820) && (RAINMARCH<=4.1380)){
    PREP=1;
}
else if ((RAINMARCH>3.3040) && (RAINMARCH<=3.5820)){
    PREP=.81;
}
else if ((RAINMARCH>3.0260) && (RAINMARCH<=3.3040)){
    PREP=.62;
}
else if (RAINMARCH<=3.0260){
    PREP=.43;
}
else if (SREG==2){
    if (currentDATE==60){
        MCRAIN2[3]=GetStochasticNumber(RAINDEVIATION2[3],
        RAINAVERAGE2[3]);
        RAINMARCH=MCRAIN2[3];
    }
    if (RAINMARCH>4.4670){
        PREP=1.57;
    }
    else if ((RAINMARCH>4.1980) && (RAINMARCH<=4.4670)){
        PREP=1.38;
    }
    else if ((RAINMARCH>3.9290) && (RAINMARCH<=4.1980)){
        PREP=1.19;
    }
    else if ((RAINMARCH>3.3910) && (RAINMARCH<=3.9290)){
        PREP=1;
    }
    else if ((RAINMARCH>3.1220) && (RAINMARCH<=3.3910)){
        PREP=.81;
    }
    else if ((RAINMARCH>2.8530) && (RAINMARCH<=3.1220)){
        PREP=.62;
    }
    else if (RAINMARCH<=2.8530){
        PREP=.43;
    }
}
else if (SREG==3){
    if (currentDATE==60){
        MCRAIN3[3]=GetStochasticNumber(RAINDEVIATION3[3],
        RAINAVERAGE3[3]);
        RAINMARCH=MCRAIN3[3];
    }
    if (RAINMARCH>4.1980){
        PREP=1.38;
    }
    else if ((RAINMARCH>3.9290) && (RAINMARCH<=4.1980)){
        PREP=1.19;
    }
    else if ((RAINMARCH>3.3910) && (RAINMARCH<=3.9290)){
        PREP=1;
    }
    else if ((RAINMARCH>3.1220) && (RAINMARCH<=3.3910)){
        PREP=.81;
    }
    else if ((RAINMARCH>2.8530) && (RAINMARCH<=3.1220)){
        PREP=.62;
    }
    else if (RAINMARCH<=2.8530){
        PREP=.43;
    }
}
RAINAVERAGE3[3];
RAINMARCH=MCRAIN3[3];
}
if (RAINMARCH>5.8240){
PREP=1.57;
}
else if ((RAINMARCH>5.0260) && (RAINMARCH<=5.8240)){
PREP=1.38;
}
else if ((RAINMARCH>4.2280) && (RAINMARCH<=5.0260)){
PREP=1.19;
}
else if ((RAINMARCH>2.6320) && (RAINMARCH<=4.2280)){
PREP=1;
}
else if ((RAINMARCH>1.8340) && (RAINMARCH<=2.6320)){
PREP=.81;
}
else if ((RAINMARCH>1.0360) && (RAINMARCH<=1.8340)){
PREP=.62;
}
else if (RAINMARCH<=1.0360){
PREP=.43;
}
}
else if ((currentDATE>=91) && (currentDATE<121)){
if (SREG==1){
if (currentDATE==91){
MCRAIN1[4]=GetStochasticNumber(RAINDEVIATION1[4],
RAINAVERAGE1[4]);
RAINAPRIL=MCRAIN1[4];
}
if (RAINAPRIL>3.6950){
PREP=1.57;
}
else if ((RAINAPRIL>3.460) && (RAINAPRIL<=3.6950)){
PREP=1.38;
}
else if ((RAINAPRIL>3.2250) && (RAINAPRIL<=3.460)){
PREP=1.19;
}
else if ((RAINAPRIL>2.7550) && (RAINAPRIL<=3.2250)){
PREP=1;
}
else if ((RAINAPRIL>2.5200) && (RAINAPRIL<=2.7550)){
PREP=.81;
}
else if ((RAINAPRIL>2.2850) && (RAINAPRIL<=2.5200)){
PREP=.62;
}
else if (RAINAPRIL<=2.2850){
    PREP=.43;
}
}
else if (SREG==2){
    if (currentDATE==91){
        MCRAIN2[4]=GetStochasticNumber(RAINDEVIATION2[4],
                                         RAINAVERAGE2[4]);
        RAINAPRIL=MCRAIN2[4];
    }
    if (RAINAPRIL>4.0150){
        PREP=1.57;
    }
    else if ((RAINAPRIL>3.7600) && (RAINAPRIL<=4.0150)){
        PREP=1.38;
    }
    else if ((RAINAPRIL>3.5050) && (RAINAPRIL<=3.7600)){
        PREP=1.19;
    }
    else if ((RAINAPRIL>2.9950) && (RAINAPRIL<=3.5050)){
        PREP=1;
    }
    else if ((RAINAPRIL>2.7400) && (RAINAPRIL<=2.9950)){
        PREP=.81;
    }
    else if ((RAINAPRIL>2.4850) && (RAINAPRIL<=2.7400)){
        PREP=.62;
    }
    else if (RAINAPRIL<=2.4850){
        PREP=.43;
    }
}
else if (SREG==3){
    if (currentDATE==91){
        MCRAIN3[4]=GetStochasticNumber(RAINDEVIATION3[4],
                                         RAINAVERAGE3[4]);
        RAINAPRIL=MCRAIN3[4];
    }
    if (RAINAPRIL>5.527){
        PREP=1.57;
    }
    else if ((RAINAPRIL>4.778) && (RAINAPRIL<=5.527)){
        PREP=1.38;
    }
    else if ((RAINAPRIL>4.029) && (RAINAPRIL<=4.778)){
        PREP=1.19;
    }
    else if ((RAINAPRIL>2.5310) && (RAINAPRIL<=4.029)){
        PREP=1;
    }
    else if ((RAINAPRIL>1.782) && (RAINAPRIL<=2.531)){
        PREP=.81;
    }
}
PREP=.81;
} else if ((RAINAPRIL>1.033) && (RAINAPRIL<=1.782)) {
PREP=.62;
} else if (RAINAPRIL<=1.033) {
PREP=.43;
} } 
else if ((currentDATE>=121) && (currentDATE<152)) {
if (SREG==1) {
if (currentDATE==121) {
MCRAIN1[5]=GetStochasticNumber(RAINDEVIATION1[5],
RAINAVERAGE1[5]);
RAINMAY=MCRAIN1[5];
}
if (RAINMAY>5.0770) {
PREP=1.57;
} else if ((RAINMAY>4.6880) && (RAINMAY<=5.0770)) {
PREP=1.38;
} else if ((RAINMAY>4.2990) && (RAINMAY<=4.6880)) {
PREP=1.19;
} else if ((RAINMAY>3.5210) && (RAINMAY<=4.2990)) {
PREP=1;
} else if ((RAINMAY>3.1320) && (RAINMAY<=3.5210)) {
PREP=.81;
} else if ((RAINMAY>2.7430) && (RAINMAY<=3.1320)) {
PREP=.62;
} else if (RAINMAY<=2.7430) {
PREP=.43;
} } else if (SREG==2) {
if (currentDATE==121) {
MCRAIN2[5]=GetStochasticNumber(RAINDEVIATION2[5],
RAINAVERAGE2[5]);
RAINMAY=MCRAIN2[5];
}
if (RAINMAY>5.0450) {
PREP=1.57;
} else if ((RAINMAY>4.7500) && (RAINMAY<=5.0450)) {
PREP=1.38;
}
else if ((RAINMAY>4.4550) && (RAINMAY<=4.7500)) {
    PREP=1.19;
}
else if ((RAINMAY>3.8650) && (RAINMAY<=4.4550)) {
    PREP=1;
}
else if ((RAINMAY>3.5700) && (RAINMAY<=3.8650)) {
    PREP=.81;
}
else if ((RAINMAY>3.2750) && (RAINMAY<=3.5700)) {
    PREP=.62;
}
else if (RAINMAY<=3.2750) {
    PREP=.43;
}
else if (SREG==3) {
    if (currentDATE==121) {
        MCRAIN3[5]=GetStochasticNumber(RAINDEVIATION3[5],
    RAINAVERAGE3[5]);
        RAINMAY=MCRAIN3[5];
    }
    if (RAINMAY>6.3060) {
        PREP=1.57;
    }
    else if ((RAINMAY>5.5240) && (RAINMAY<=6.3060)) {
        PREP=1.38;
    }
    else if ((RAINMAY>4.7420) && (RAINMAY<=5.5240)) {
        PREP=1.19;
    }
    else if ((RAINMAY>3.1780) && (RAINMAY<=4.7420)) {
        PREP=1;
    }
    else if ((RAINMAY>2.3960) && (RAINMAY<=3.1780)) {
        PREP=.81;
    }
    else if ((RAINMAY>1.6140) && (RAINMAY<=2.3960)) {
        PREP=.62;
    }
    else if (RAINMAY<=1.6140) {
        PREP=.43;
    }
}
}
else if ((currentDATE>=152) && (currentDATE<182)) {
    if (SREG==1) {
        if (currentDATE==152) {
            MCRAIN1[6]=GetStochasticNumber(RAINDEVIATION1[6],
        RAINAVERAGE1[6]);
            RAINJUNE=MCRAIN1[6];
        }
    }
if (RAINJUNE>5.0190)
    PREP=1.57;
else if ((RAINJUNE>4.5960) && (RAINJUNE<=5.0190))
    PREP=1.38;
else if ((RAINJUNE>4.1730) && (RAINJUNE<=4.5960))
    PREP=1.19;
else if ((RAINJUNE>3.3270) && (RAINJUNE<=4.1730))
    PREP=1;
else if ((RAINJUNE>2.9040) && (RAINJUNE<=3.3270))
    PREP=.81;
else if ((RAINJUNE>2.4810) && (RAINJUNE<=2.9040))
    PREP=.62;
else if (RAINJUNE<=2.4810)
    PREP=.43;
else if (SREG==2)
    if (currentDATE==152)
        MCRAIN2[6]=GetStochasticNumber(RAINDEVIAION2[6],
            RAINAVERAGE2[6]);
    RAINJUNE=MCRAIN2[6];
    if (RAINJUNE>4.5260)
        PREP=1.57;
    else if ((RAINJUNE>4.2040) && (RAINJUNE<=4.5260))
        PREP=1.38;
    else if ((RAINJUNE>3.8820) && (RAINJUNE<=4.2040))
        PREP=1.19;
    else if ((RAINJUNE>3.2380) && (RAINJUNE<=3.8820))
        PREP=1;
    else if ((RAINJUNE>2.9160) && (RAINJUNE<=3.2380))
        PREP=.81;
    else if ((RAINJUNE>2.5940) && (RAINJUNE<=2.9160))
        PREP=.62;
    else if (RAINJUNE<=2.5940)
        PREP=.43;
else if (SREG==3){
    if (currentDATE==152){
        MCRAIN3[6]=GetStochasticNumber(RAINDEVIATION3[6],
             RAINAVERAGE3[6]);
        RAINJUNE=MCRAIN3[6];
    }
    if (RAINJUNE>5.7630){
        PREP=1.57;
    }
    else if ((RAINJUNE>5.0020) && (RAINJUNE<=5.7630)){
        PREP=1.38;
    }
    else if ((RAINJUNE>4.2410) && (RAINJUNE<=5.0020)){
        PREP=1.19;
    }
    else if ((RAINJUNE>2.7190) && (RAINJUNE<=4.2410)){
        PREP=1;
    }
    else if ((RAINJUNE>1.9580) && (RAINJUNE<=2.7190)){
        PREP=.81;
    }
    else if ((RAINJUNE>1.1970) && (RAINJUNE<=1.9580)){
        PREP=.62;
    }
    else if (RAINJUNE<=1.1970){
        PREP=.43;
    }
}
else if ((currentDATE>=182) && (currentDATE<213)){
    if (SREG==1){
        if (currentDATE==182){
            MCRAIN1[7]=GetStochasticNumber(RAINDEVIATION1[7],
                 RAINAVERAGE1[7]);
            RAINJULY=MCRAIN1[7];
        }
        if (RAINJULY>6.3040){
            PREP=1.57;
        }
        else if ((RAINJULY>5.7060) && (RAINJULY<=6.3040)){
            PREP=1.38;
        }
        else if ((RAINJULY>5.1080) && (RAINJULY<=5.7060)){
            PREP=1.19;
        }
        else if ((RAINJULY>3.9120) && (RAINJULY<=5.1080)){
            PREP=1;
        }
        else if ((RAINJULY>3.3140) && (RAINJULY<=3.9120)){
            PREP=.81;
        }
    }
else if ((RAINJULY>2.7160) && (RAINJULY<=3.3140)){
  PREP=.62;
}
else if (RAINJULY<=2.7160){
  PREP=.43;
}
else if (SREG==2){
  if (currentDATE==182){
    MCRAIN2[7]=GetStochasticNumber(RAINDEVIATION2[7],
      RAINAVERAGE2[7]);
    RAINJULY=MCRAIN2[7];
  }
  if (RAINJULY>5.2680){
    PREP=1.57;
  }
  else if ((RAINJULY>4.9220) && (RAINJULY<=5.2680)){
    PREP=1.38;
  }
  else if ((RAINJULY>4.5760) && (RAINJULY<=4.9220)){
    PREP=1.19;
  }
  else if ((RAINJULY>3.8840) && (RAINJULY<=4.5760)){
    PREP=1;
  }
  else if ((RAINJULY>3.5380) && (RAINJULY<=3.8840)){
    PREP=.81;
  }
  else if ((RAINJULY>3.1920) && (RAINJULY<=3.5380)){
    PREP=.62;
  }
  else if (RAINJULY<=3.1920){
    PREP=.43;
  }
}
else if (SREG==3){
  if (currentDATE==182){
    MCRAIN3[7]=GetStochasticNumber(RAINDEVIATION3[7],
      RAINAVERAGE3[7]);
    RAINJULY=MCRAIN3[7];
  }
  if (RAINJULY>6.5130){
    PREP=1.57;
  }
  else if ((RAINJULY>5.6820) && (RAINJULY<=6.5130)){
    PREP=1.38;
  }
  else if ((RAINJULY>4.8510) && (RAINJULY<=5.6820)){
    PREP=1.19;
  }
  else if ((RAINJULY>3.1890) && (RAINJULY<=4.8510)){
    PREP=.81;
  }
  else if ((RAINJULY>2.6820) && (RAINJULY<=3.1890)){
    PREP=.62;
  }
  else if (RAINJULY<=2.6820){
    PREP=.43;
  }
}
PREP=1;
}
else if ((RAINJULY>2.3580) && (RAINJULY<=3.1890)){
  PREP=.81;
}
else if ((RAINJULY>1.5270) && (RAINJULY<=2.3580)){
  PREP=.62;
}
else if (RAINJULY<=1.5270){
  PREP=.43;
}
}

else if ((currentDATE==213) && (currentDATE<244)){
  if (SREG==1){
    if (currentDATE==213){
      MCRAIN1[8]=GetStochasticNumber(RAINDEVIATION1[8], RAINAVERAGE1[8]);
      RAINAUGUST=MCRAIN1[8];
      if (RAINAUGUST>6.3370){
        PREP=1.57;
      }
      else if ((RAINAUGUST>5.7280) && (RAINAUGUST<=6.3370)){
        PREP=1.38;
      }
      else if ((RAINAUGUST>5.1190) && (RAINAUGUST<=5.7280)){
        PREP=1.19;
      }
      else if ((RAINAUGUST>3.9010) && (RAINAUGUST<=5.1190)){
        PREP=1;
      }
      else if ((RAINAUGUST>3.2920) && (RAINAUGUST<=3.9010)){
        PREP=.81;
      }
      else if ((RAINAUGUST>2.6830) && (RAINAUGUST<=3.2920)){
        PREP=.62;
      }
      else if (RAINAUGUST<=2.6830){
        PREP=.43;
      }
    }
    else if (SREG==2){
      if (currentDATE==213){
        MCRAIN2[8]=GetStochasticNumber(RAINDEVIATION2[8], RAINAVERAGE2[8]);
        RAINAUGUST=MCRAIN2[8];
        if (RAINAUGUST>5.0880){
          PREP=1.57;
        }
      }
    }
  }
}
else if ((RAIN\textsc{august}>4.7420) \&\& (RAIN\textsc{august}<=5.0880))
  PREP=1.38;
} 
else if ((RAIN\textsc{august}>4.3960) \&\& (RAIN\textsc{august}<=4.7420))
  PREP=1.19;
} 
else if ((RAIN\textsc{august}>3.7040) \&\& (RAIN\textsc{august}<=4.3960))
  PREP=1;
} 
else if ((RAIN\textsc{august}>3.3580) \&\& (RAIN\textsc{august}<=3.7040))
  PREP=.81;
} 
else if ((RAIN\textsc{august}>3.0120) \&\& (RAIN\textsc{august}<=3.3580))
  PREP=.62;
} 
else if (RAIN\textsc{august}<=3.0120)
  PREP=.43;
} 
} 
else if (S\textsc{reg}=3){
  if (current\textsc{date}==213){
    MC\textsc{rain}3[8]=GetStochasticNumber(RAIN\textsc{deviation}3[8],
    RAIN\textsc{average}3[8]);
    RAIN\textsc{august}=MC\textsc{rain}3[8];
    if (RAIN\textsc{august}>5.7980)
      PREP=1.57;
    } 
else if ((RAIN\textsc{august}>5.0620) \&\& (RAIN\textsc{august}<=5.7980))
      PREP=1.38;
    } 
else if ((RAIN\textsc{august}>4.3260) \&\& (RAIN\textsc{august}<=5.0620))
      PREP=1.19;
    } 
else if ((RAIN\textsc{august}>2.8540) \&\& (RAIN\textsc{august}<=4.3260))
      PREP=1;
    } 
else if ((RAIN\textsc{august}>2.1180) \&\& (RAIN\textsc{august}<=2.8540))
      PREP=.81;
    } 
else if ((RAIN\textsc{august}>1.3820) \&\& (RAIN\textsc{august}<=2.1180))
      PREP=.62;
    } 
else if (RAIN\textsc{august}<=1.3820)
      PREP=.43;
    } 
} 
} 
else if ((current\textsc{date}==244) \&\& (current\textsc{date}<274))
  if (S\textsc{reg}=1){
    if (current\textsc{date}==244){

MCRAIN1[9]=GetStochasticNumber(RAINDEVIATION1[9], RAINAVERAGE1[9]);
RAINSEPTEMBER=MCRAIN1[9];
}
if (RAINSEPTEMBER>5.2780){
PREP=1.57;
}
else if ((RAINSEPTEMBER>4.8020) && (RAINSEPTEMBER<=5.2780)){
PREP=1.38;
}
else if ((RAINSEPTEMBER>4.3260) && (RAINSEPTEMBER<=4.8020)){
PREP=1.19;
}
else if ((RAINSEPTEMBER>3.3740) && (RAINSEPTEMBER<=4.3260)){
PREP=1;
}
else if ((RAINSEPTEMBER>2.8980) && (RAINSEPTEMBER<=3.3740)){
PREP=.81;
}
else if ((RAINSEPTEMBER>2.4220) && (RAINSEPTEMBER<=2.8980)){
PREP=.62;
}
else if (RAINSEPTEMBER<=2.4220){
PREP=.43;
}
else if (SREG==2){
if (currentDATE==244){
MCRAIN2[9]=GetStochasticNumber(RAINDEVIATION2[9], RAINAVERAGE2[9]);
RAINSEPTEMBER=MCRAIN2[9];
}
if (RAINSEPTEMBER>4.5640){
PREP=1.57;
}
else if ((RAINSEPTEMBER>4.2260) && (RAINSEPTEMBER<=4.5640)){
PREP=1.38;
}
else if ((RAINSEPTEMBER>3.3880) && (RAINSEPTEMBER<=4.2260)){
PREP=1.19;
}
else if ((RAINSEPTEMBER>3.2120) && (RAINSEPTEMBER<=3.3880)){
PREP=1;
}
else if ((RAINSEPTEMBER>2.8740) && (RAINSEPTEMBER<=3.2120)){
PREP=.81;
}
else if ((RAINSEPTEMBER>2.5360) && (RAINSEPTEMBER<=2.8740)){
PREP=.62;
}
else if (RAINSEPTEMBER<=2.5360){
PREP=.43;
PREP = 0.43;
}
}
else if (SREG==3) {
if (currentDATE==244) {
MCRAIN3[9]=GetStochasticNumber(RAINDEVIATION3[9],
RAINAVG3E[9]);
RAINSEPTEMBER=MCRAIN3[9];
}
if (RAINSEPTEMBER>5.7460) {
PREP=1.57;
}
else if ((RAINSEPTEMBER>4.9740) && (RAINSEPTEMBER<=5.7460)) {
PREP=1.38;
}
else if ((RAINSEPTEMBER>4.2020) && (RAINSEPTEMBER<=4.9740)) {
PREP=1.19;
}
else if ((RAINSEPTEMBER>2.6580) && (RAINSEPTEMBER<=4.2020)) {
PREP=1;
}
else if ((RAINSEPTEMBER>1.8860) && (RAINSEPTEMBER<=2.6580)) {
PREP=.81;
}
else if ((RAINSEPTEMBER>1.1140) && (RAINSEPTEMBER<=1.8860)) {
PREP=.62;
}
else if (RAINSEPTEMBER<=1.1140) {
PREP=.43;
}
}
else if ((currentDATE==274) && (currentDATE<305)) {
if (SREG==1) {
if (currentDATE==274) {
MCRAIN1[10]=GetStochasticNumber(RAINDEVIATION1[10],
RAINAVG1[10]);
RAINOCTOBER=MCRAIN1[10];
}
if (RAINOCTOBER>4.3850) {
PREP=1.57;
}
else if ((RAINOCTOBER>4.0200) && (RAINOCTOBER<=4.3850)) {
PREP=1.38;
}
else if ((RAINOCTOBER>3.655) && (RAINOCTOBER<=4.0200)) {
PREP=1.19;
}
else if ((RAINOCTOBER>2.925) && (RAINOCTOBER<=3.655)) {
PREP=1;
}
}
else if ((RAINOCTOBER>2.560) && (RAINOCTOBER<=2.925))
    PREP=.81;
} else if ((RAINOCTOBER>2.195) && (RAINOCTOBER<=2.560))
    PREP=.62;
} else if (RAINOCTOBER<=2.195)
    PREP=.43;
} else if (SREG==2){
    if (currentDATE==274){
        MCRAIN2[10]=GetStochasticNumber(RAINDEVIATION2[10],
            RAINAVERAGE2[10]);
        RAINOCTOBER=MCRAIN2[10];
    }
    if (RAINOCTOBER>5.1820){
        PREP=1.57;
    } else if ((RAINOCTOBER>4.7480) && (RAINOCTOBER<=5.1820)){
        PREP=1.38;
    } else if ((RAINOCTOBER>4.3140) && (RAINOCTOBER<=4.7480)){
        PREP=1.19;
    } else if ((RAINOCTOBER>3.446) && (RAINOCTOBER<=4.3140)){
        PREP=1;
    } else if ((RAINOCTOBER>3.012) && (RAINOCTOBER<=3.446)){
        PREP=.81;
    } else if ((RAINOCTOBER>2.578) && (RAINOCTOBER<=3.012)){
        PREP=.62;
    } else if (RAINOCTOBER<=2.578){
        PREP=.43;
    }
} else if (SREG==3){
    if (currentDATE==274){
        MCRAIN3[10]=GetStochasticNumber(RAINDEVIATION3[10],
            RAINAVERAGE3[10]);
        RAINOCTOBER=MCRAIN3[10];
    }
    if (RAINOCTOBER>5.9420){
        PREP=1.57;
    } else if ((RAINOCTOBER>5.1480) && (RAINOCTOBER<=5.9420)){
        PREP=1.38;
    } else if ((RAINOCTOBER>4.3540) && (RAINOCTOBER<=5.1480)){
        PREP=1.19;
    } else if ((RAINOCTOBER>3.446) && (RAINOCTOBER<=4.3540)){
        PREP=1;
    } else if ((RAINOCTOBER>3.012) && (RAINOCTOBER<=3.446)){
        PREP=.81;
    } else if ((RAINOCTOBER>2.578) && (RAINOCTOBER<=3.012)){
        PREP=.62;
    } else if (RAINOCTOBER<=2.578){
        PREP=.43;
    }
}
PREP=1.19;
} else if ((RAINOCTOBER>2.7660) && (RAINOCTOBER<=4.3540)) { PREP=1;
} else if ((RAINOCTOBER>1.9720) && (RAINOCTOBER<=2.7660)) { PREP=.81;
} else if ((RAINOCTOBER>1.1780) && (RAINOCTOBER<=1.9720)) { PREP=.62;
} else if (RAINOCTOBER<=1.1780) { PREP=.43;
} } else if ((currentDATE==305) && (currentDATE<335)) { if (SREG==1) { if (currentDATE==305) { MCRAIN1[11]=GetStochasticNumber(RAINDEVIATION1[11], RAINAVERAGE1[11]); RAINNOVEMBER=MCRAIN1[11]; } if (RAINNOVEMBER>3.8540) { PREP=1.57;
} else if ((RAINNOVEMBER>3.6260) && (RAINNOVEMBER<=3.8540)) { PREP=1.38;
} else if ((RAINNOVEMBER>3.3980) && (RAINNOVEMBER<=3.6260)) { PREP=1.19;
} else if ((RAINNOVEMBER>2.9420) && (RAINNOVEMBER<=3.3980)) { PREP=1;
} else if ((RAINNOVEMBER>2.7140) && (RAINNOVEMBER<=2.9420)) { PREP=.81;
} else if ((RAINNOVEMBER>2.4860) && (RAINNOVEMBER<=2.7140)) { PREP=.62;
} else if (RAINNOVEMBER<=2.4860) { PREP=.43;
if (RAINNOVEMBER>4.4310)
  PREP=1.57;
else if ((RAINNOVEMBER>4.0940) && (RAINNOVEMBER<=4.4310))
  PREP=1.38;
else if ((RAINNOVEMBER>3.7570) && (RAINNOVEMBER<=4.0940))
  PREP=1.19;
else if ((RAINNOVEMBER>3.0830) && (RAINNOVEMBER<=3.7570))
  PREP=1;
else if ((RAINNOVEMBER>2.7460) && (RAINNOVEMBER<=3.0830))
  PREP=.81;
else if ((RAINNOVEMBER>2.4090) && (RAINNOVEMBER<=2.7460))
  PREP=.62;
else if (RAINNOVEMBER<=2.4090)
  PREP=.43;
}
else if (SREG==3){
  if (currentDATE==305)
    RAINAVERAGE3[11]);
    RAINNOVEMBER=MCRAIN3[11];
  if (RAINNOVEMBER>5.4460)
    PREP=1.57;
  else if ((RAINNOVEMBER>4.6840) && (RAINNOVEMBER<=5.4460))
    PREP=1.38;
  else if ((RAINNOVEMBER>3.9220) && (RAINNOVEMBER<=4.6840))
    PREP=1.19;
  else if ((RAINNOVEMBER>2.3980) && (RAINNOVEMBER<=3.9220))
    PREP=1;
  else if ((RAINNOVEMBER>1.6360) && (RAINNOVEMBER<=2.3980))
    PREP=.81;
  else if ((RAINNOVEMBER>.8740) && (RAINNOVEMBER<=1.6360))
    PREP=.62;
  else if (RAINNOVEMBER<=.8740)
    PREP=.43;
}
else if ((currentDATE==335) && (currentDATE<=365)){
    if (SREG==1){
        if (currentDATE==335){
            MCRAIN1[12]=GetStochasticNumber(RAINDEVIATION1[12], RAINAVERAGE1[12]);
            RAINDECEMBER=MCRAIN1[12];
        }
        if (RAINDECEMBER>4.0420){
            PREP=1.57;
        }
        else if ((RAINDECEMBER>3.7980) && (RAINDECEMBER<=4.0420)){
            PREP=1.38;
        }
        else if ((RAINDECEMBER>3.5540) && (RAINDECEMBER<=3.7980)){
            PREP=1.19;
        }
        else if ((RAINDECEMBER>3.0660) && (RAINDECEMBER<=3.5540)){
            PREP=1;
        }
        else if ((RAINDECEMBER>2.8220) && (RAINDECEMBER<=3.0660)){
            PREP=.81;
        }
        else if ((RAINDECEMBER>2.5780) && (RAINDECEMBER<=2.8220)){
            PREP=.62;
        }
        else if (RAINDECEMBER<=2.5780){
            PREP=.43;
        }
    }
    else if (SREG==2){
        if (currentDATE==335){
            MCRAIN2[12]=GetStochasticNumber(RAINDEVIATION2[12], RAINAVERAGE2[12]);
            RAINDECEMBER=MCRAIN2[12];
        }
        if (RAINDECEMBER>3.7030){
            PREP=1.57;
        }
        else if ((RAINDECEMBER>3.5220) && (RAINDECEMBER<=3.7030)){
            PREP=1.38;
        }
        else if ((RAINDECEMBER>3.3410) && (RAINDECEMBER<=3.5220)){
            PREP=1.19;
        }
        else if ((RAINDECEMBER>2.9790) && (RAINDECEMBER<=3.3410)){
            PREP=1;
        }
        else if ((RAINDECEMBER>2.7980) && (RAINDECEMBER<=2.9790)){
            PREP=.81;
        }
        else if ((RAINDECEMBER>2.6170) && (RAINDECEMBER<=2.7980)){
            PREP=0.38;
        }
        else if (RAINDECEMBER<=2.6170){
            PREP=.43;
        }
    }
}
PREP=.62;
}
else if (RAINDECEMBER<=2.6170){
  PREP=.43;
}
}
else if (SREG==3){
  if (currentDATE==335){
    MCRAIN3[12]=GetStochasticNumber(RAINDEVIATION3[12],
    RAINAVERAGE3[12]);
    RAINDECEMBER=MCRAIN3[12];
  }
  if (RAINDECEMBER>4.919){
    PREP=1.57;
  }
  else if ((RAINDECEMBER>4.246) && (RAINDECEMBER<=4.919)){
    PREP=1.38;
  }
  else if ((RAINDECEMBER>3.573) && (RAINDECEMBER<=4.246)){
    PREP=1.19;
  }
  else if ((RAINDECEMBER>2.227) && (RAINDECEMBER<=3.573)){
    PREP=1;
  }
  else if ((RAINDECEMBER>1.554) && (RAINDECEMBER<=2.227)){
    PREP=.81;
  }
  else if ((RAINDECEMBER>.8810) && (RAINDECEMBER<=1.554)){
    PREP=.62;
  }
  else if (RAINDECEMBER<=.8810){
    PREP=.43;
  }
  }
}
if ( (SREG==1) && (YEAR==-7) && (currentDATE==1) ){ MCARAIN1=GetStochasticNumber(4.58, 44.62);
  ANNRAIN=MCARAIN1;
  if (ANNRAIN>58.36){
    APREP=1.57;
  } else if ((ANNRAIN>53.78) && (ANNRAIN<=58.36)){
    APREP=1.38;
  } else if ((ANNRAIN>49.20) && (ANNRAIN<=53.78)){
    APREP=1.19;
  } else if ((ANNRAIN>40.04) && (ANNRAIN<=49.20)){
    APREP=1;
  } else if ((ANNRAIN>35.46) && (ANNRAIN<=40.04)){

APREP = .81;
}
else if ((ANNRAIN > 30.88) && (ANNRAIN <= 35.46)) {
    APREP = .62;
}
else if (ANNRAIN <= 30.88) {
    APREP = .43;
}
else if ((SREG == 2) && (YEAR == -7) && (currentDATE == 1)) {
    MCARAIN2 = GetStochasticNumber(3.723, 43.22);
    ANNRAIN = MCARAIN2;
    if (ANNRAIN > 54.389) {
        APREP = 1.57;
    }
    else if ((ANNRAIN > 50.666) && (ANNRAIN <= 54.389)) {
        APREP = 1.38;
    }
    else if ((ANNRAIN > 46.943) && (ANNRAIN <= 50.666)) {
        APREP = 1.19;
    }
    else if ((ANNRAIN > 43.22) && (ANNRAIN <= 46.943)) {
        APREP = 1;
    }
    else if ((ANNRAIN > 39.497) && (ANNRAIN <= 43.22)) {
        APREP = .81;
    }
    else if ((ANNRAIN > 35.774) && (ANNRAIN <= 39.497)) {
        APREP = .62;
    }
    else if (ANNRAIN <= 35.774) {
        APREP = .43;
    }
}
else if ((SREG == 3) && (YEAR == -7) && (currentDATE == 1)) {
    MCARAIN3 = GetStochasticNumber(5.976, 41.41);
    ANNRAIN = MCARAIN3;
    if (ANNRAIN > 59.338) {
        APREP = 1.57;
    }
    else if ((ANNRAIN > 53.362) && (ANNRAIN <= 59.338)) {
        APREP = 1.38;
    }
    else if ((ANNRAIN > 47.386) && (ANNRAIN <= 53.362)) {
        APREP = 1.19;
    }
    else if ((ANNRAIN > 35.434) && (ANNRAIN <= 47.386)) {
        APREP = 1;
    }
    else if ((ANNRAIN > 35.434) && (ANNRAIN <= 35.434)) {
        APREP = .81;
    }
else if ((ANNRAIN>29.458) & (ANNRAIN<=35.434)){
    APREP=.62;
}
else if (ANNRAIN<=29.458){
    APREP=.43;
}

if ((YEAR == -7) && (currentDATE==1)){
    for (i=0; i<21; i++){
        PAVAIL+=((FAVAIL[i]*FR[i])*.45359);
    }
    for (i=1; i<=pasture; i++){
        permPAST+=TPAS[i];
        permPAST+=HLAND[i];
        permRPAST+=TRPAS[i];
        permRPAST+=HRLAND[i];
    }
    CurrentSIZE=permPAST;
    CurrentSIZES=permRPAST;
    PAVAIL = ( (PAVAIL * (CurrentSIZE+CurrentSIZES)) * (SOIL*PMAN*SLOP*WEED*PMAIN*APREP) ) ;
}

DaySen=0;
DaySen=PAVAIL* SENE*2.2049;
TotAnnSen+=DaySen;
PAVAIL =((PAVAIL ) * (1 - SENE));
if (PAVAIL<0){
    PAVAIL=0;
}

if ((YEAR == -7) && (currentDATE==1)){
    for (i=0; i<22; i++){
        FAME+=FME[i]*FR[i];
        FANEM+=FNEM[i]*FR[i];
        FANEG+=FNEG[i]*FR[i];
        TDNP+=FTDN[i]*FR[i];
        mainFDM+=hayFDM[i]*FR[i];
    }
}

if ((currentDATE>=1) && (currentDATE<32)){
    permPAST=0;
    hayLANDS=0;
    woodLANDS=0;
    for (i=1; i<=pasture; i++){
        if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
            permPAST+=TPAS[i];
            hayLANDS+=HLAND[i];
            woodLANDS+=WLAND[i];
        }else if ( (currentDATE<dateOUT[i]) || (current-
DATE>=dateIN[i])
permPAST+=TPAS[i];
hayLANDS+= 0;
woodLANDS+=WLAND[i];
permPAST+=HLAND[i];
}

CurrentSIZE=permPAST;
}
else if ((currentDATE>=32) && (currentDATE<60))
permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) )
permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
}
else if ( (currentDATE<dateOUT[i]) || (currentDATE=dateIN[i]) )
permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
}
CurrentSIZE=permPAST;
}
else if ((currentDATE>=60) && (currentDATE<91))
permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) )
permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
}
else if ( (currentDATE<dateOUT[i]) || (currentDATE=dateIN[i]) )
permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
}
CurrentSIZE=permPAST;
}
else if ((currentDATE>=91) && (currentDATE<121))
permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
    if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
        permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
    } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
        permPAST+=TPAS[i];
hayLANDS+=0;
woodLANDS+=WLAND[i];
        permPAST+=HLAND[i];
    }
}
CurrentSIZE=permPAST;
}
else if (((currentDATE>=121) && (currentDATE<152))){
    permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
    if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
        permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
    } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
        permPAST+=TPAS[i];
hayLANDS+=0;
woodLANDS+=WLAND[i];
        permPAST+=HLAND[i];
    }
}
CurrentSIZE=permPAST;
}
else if ((currentDATE>=152) && (currentDATE<182)){
    permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
    if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
        permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
    } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
        permPAST+=TPAS[i];
    }
}
else if ((currentDATE>=182) && (currentDATE<212)){
    permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
    if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
        permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
    } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
        permPAST+=TPAS[i];
    }
}
else if ((currentDATE>=212) && (currentDATE<242)){
    permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
    if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
        permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
    } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
        permPAST+=TPAS[i];
    }
}
hayLANDS+= 0;
woodLANDS+=WLAND[i];
permPAST+=HLAND[i];
}
}
CurrentSIZE=permPAST;
}
else if ((currentDATE>=182) && (currentDATE<213)){
permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
  if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
    permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
  }
  else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
    permPAST+=TPAS[i];
hayLANDS+= 0;
woodLANDS+=WLAND[i];
permPAST+=HLAND[i];
  }
}
CurrentSIZE=permPAST;
}
else if (((currentDATE>=213) && (currentDATE<244))){
permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<=pasture; i++){
  if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
    permPAST+=TPAS[i];
hayLANDS+=HLAND[i];
woodLANDS+=WLAND[i];
  }
  else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
    permPAST+=TPAS[i];
hayLANDS+= 0;
woodLANDS+=WLAND[i];
permPAST+=HLAND[i];
  }
}
CurrentSIZE=permPAST;
}
else if ((currentDATE>=244) && (currentDATE<274)){
permPAST=0;
hayLANDS=0;
woodLANDS=0;
for (i=1; i<pasture; i++){
    if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
        permPAST+=TPAS[i];
        hayLANDS+=HLAND[i];
        woodLANDS+=WLAND[i];
    } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
        permPAST+=TPAS[i];
        hayLANDS+=0;
        woodLANDS+=WLAND[i];
        permPAST+=HLAND[i];
    }
    CurrentSIZE=permPAST;
}
else if ((currentDATE>=274) && (currentDATE<305)){
    permPAST=0;
    hayLANDS=0;
    woodLANDS=0;
    for (i=1; i<pasture; i++){
        if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
            permPAST+=TPAS[i];
            hayLANDS+=HLAND[i];
            woodLANDS+=WLAND[i];
        } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
            permPAST+=TPAS[i];
            hayLANDS+=0;
            woodLANDS+=WLAND[i];
            permPAST+=HLAND[i];
        }
    }
    CurrentSIZE=permPAST;
}
else if ((currentDATE>=305) && (currentDATE<335)){
    permPAST=0;
    hayLANDS=0;
    woodLANDS=0;
    for (i=1; i<pasture; i++){
        if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ){
            permPAST+=TPAS[i];
            hayLANDS+=HLAND[i];
            woodLANDS+=WLAND[i];
        } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ){
            permPAST+=TPAS[i];
            hayLANDS+=0;
            woodLANDS+=WLAND[i];
        }
    }
    CurrentSIZE=permPAST;
}
permPAST+=HLAND[i];
}
}
CurrentSIZE=permPAST;
}
else if ((currentDATE>=335) && (currentDATE<=365)) {
    permPAST=0;
    hayLANDS=0;
    woodLANDS=0;
    for (i=1; i<=pasture; i++) {
        if ( (currentDATE>=dateOUT[i]) && (currentDATE<dateIN[i]) ) {
            permPAST+=TPAS[i];
            hayLANDS+=HLAND[i];
            woodLANDS+=WLAND[i];
        } else if ( (currentDATE<dateOUT[i]) || (currentDATE>=dateIN[i]) ) {
            permPAST+=TPAS[i];
            hayLANDS+=0;
            woodLANDS+=WLAND[i];
            permPAST+=HLAND[i];
        }
    }
    CurrentSIZE=permPAST;
}
else if ((currentDATE>=1) && (currentDATE<32)) {
    permRPAST=0;
    hayRLANDS=0;
    woodRLANDS=0;
    for (i=1; i<=rpasture; i++) {
        if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ) {
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        } else if ( (currentDATE<dateROUT[i]) || (currentDATE>=dateRIN[i]) ) {
            permRPAST+=TRPAS[i];
            hayRLANDS+=0;
            woodRLANDS+=WRLAND[i];
            permRPAST+=HRLAND[i];
        }
    }
    CurrentSIZES=permRPAST;
}
else if ((currentDATE>=32) && (currentDATE<60)) {
    permRPAST=0;
    hayRLANDS=0;
    woodRLANDS=0;
}
for (i=1; i<=rpasture; i++){
    if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ){
        permRPAST+=TRPAS[i];
        hayRLANDS+=HRLAND[i];
        woodRLANDS+=WRLAND[i];
    } else if ( (currentDATE<dateROUT[i]) || (currentDATE>=dateRIN[i]) ){
        permRPAST+=TRPAS[i];
        hayRLANDS= 0;
        woodRLANDS+=WRLAND[i];
        permRPAST+=HRLAND[i];
    }
    CurrentSIZES=permRPAST;
}
else if (((currentDATE>=60) && (currentDATE<91))){
    permRPAST=0;
    hayRLANDS=0;
    woodRLANDS=0;
    for (i=1; i<=rpasture; i++){
        if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ){
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        } else if ( (currentDATE<dateROUT[i]) || (currentDATE>=dateRIN[i]) ){
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        }
    }
    CurrentSIZES=permRPAST;
}
else if (((currentDATE>=91) && (currentDATE<121))){
    permRPAST=0;
    hayRLANDS=0;
    woodRLANDS=0;
    for (i=1; i<=rpasture; i++){
        if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ){
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        } else if ( (currentDATE<dateROUT[i]) || (currentDATE>=dateRIN[i]) ){
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        }
    }
    CurrentSIZES=permRPAST;
}
permRPAST+=TRPAS[i];
hayRLANDS+= 0;
woodRLANDS+=WRLAND[i];
permRPAST+=HRLAND[i];
}
}
CurrentSIZES=permRPAST;
}
else if ( (currentDATE>121) && (currentDATE<152))
permRPAST=0;
hayRLANDS=0;
woodRLANDS=0;
for (i=1; i<=rpasture; i++){
if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) )
permRPAST+=TRPAS[i];
hayRLANDS+=HRLAND[i];
woodRLANDS+=WRLAND[i];
}
else if ( (currentDATE<dateROUT[i]) || (currentDATE>=dateRIN[i]) )
permRPAST+=TRPAS[i];
hayRLANDS+=HRLAND[i];
woodRLANDS+=WRLAND[i];
}
CurrentSIZES=permRPAST;
}
else if ((currentDATE>=152) && (currentDATE<182))
permRPAST=0;
hayRLANDS=0;
woodRLANDS=0;
for (i=1; i<=rpasture; i++){
if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) )
permRPAST+=TRPAS[i];
hayRLANDS+=HRLAND[i];
woodRLANDS+=WRLAND[i];
}
else if ( (currentDATE<dateROUT[i]) || (currentDATE>=dateRIN[i]) )
permRPAST+=TRPAS[i];
hayRLANDS+=HRLAND[i];
woodRLANDS+=WRLAND[i];
}
else if ((currentDATE>=182) && (currentDATE<213))
permRPAST=0;
hayRLANDS=0;
woodRLANDS=0;
for (i=1; i<=rpasture; i++){
if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) )
permRPAST+=TRPAS[i];
hayRLANDS+=HRLAND[i];
woodRLANDS+=WRLAND[i];
}
else if ( (currentDATE<dateROUT[i]) || (currentDATE>=dateRIN[i]) )
permRPAST+=TRPAS[i];
hayRLANDS+= 0;
woodRLANDS+=WRLAND[i];
permRPAST+=HRLAND[i];
}
CurrentSIZES=permRPAST;
}
else if ( ((currentDATE)>=182) && (currentDATE<213)){
permRPAST=0;
hayRLANDS=0;
woodRLANDS=0;
for (i=1; i<=rpasture; i++){
  if ( (currentDATE=\geq dateROUT[i]) && (currentDATE\prec dateRIN[i]) ){
    permRPAST+=TRPAS[i];
hayRLANDS+=HRLAND[i];
woodRLANDS+=WRLAND[i];
  }
else if ( (currentDATE\geq dateROUT[i]) || (currentDATE\prec dateRIN[i]) ){
    permRPAST+=TRPAS[i];
hayRLANDS+=0;
woodRLANDS+=WRLAND[i];
    permRPAST+=HRLAND[i];
  }
}
CurrentSIZES=permRPAST;
}
else if ((currentDATE\geq 213) && (currentDATE\prec 244)){
permRPAST=0;
hayRLANDS=0;
woodRLANDS=0;
for (i=1; i<=rpasture; i++){
  if ( (currentDATE=\geq dateROUT[i]) && (currentDATE\prec dateRIN[i]) ){
    permRPAST+=TRPAS[i];
hayRLANDS+=HRLAND[i];
woodRLANDS+=WRLAND[i];
  }
else if ( (currentDATE\prec dateROUT[i]) || (currentDATE\geq dateRIN[i]) ){
    permRPAST+=TRPAS[i];
hayRLANDS+= 0;
woodRLANDS+=WRLAND[i];
permRPAST+=HRLAND[i];
  }
}
CurrentSIZES=permRPAST;
}
else if ((currentDATE\geq 244) && (currentDATE\prec 274)){
permRPAST=0;
hayRLANDS=0;
woodRLANDS=0;
for (i=1; i<=rpasture; i++){
  if ( (currentDATE=\geq dateROUT[i]) && (currentDATE\prec dateRIN[i]) ){
    permRPAST+=TRPAS[i];
hayRLANDS+=HRLAND[i];
woodRLANDS+=WRLAND[i];
  }
}

else if ( (currentDATE<dateROUT[i]) || (currentDATE=dateRIN[i]) ){
    permRPAST+=TRPAS[i];
    hayRLANDS+= 0;
    woodRLANDS+=WRLAND[i];
    permRPAST+=HRLAND[i];
}
CurrentSIZES=permRPAST;
}
else if ((currentDATE>=274) && (currentDATE<305)){
    permRPAST=0;
    hayRLANDS=0;
    woodRLANDS=0;
    for (i=1; i<=rpasture; i++){
        if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ){
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        }
    }
    CurrentSIZES=permRPAST;
}
else if ((currentDATE<dateROUT[i]) || (currentDATE<305)){
    permRPAST=0;
    hayRLANDS=0;
    woodRLANDS=0;
    for (i=1; i<=rpasture; i++){
        if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ){
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        }
    }
    CurrentSIZES=permRPAST;
}
else if ((currentDATE>=305) && (currentDATE<335)){
    permRPAST=0;
    hayRLANDS=0;
    woodRLANDS=0;
    for (i=1; i<=rpasture; i++){
        if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ){
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        }
    }
    CurrentSIZES=permRPAST;
}
else if ((currentDATE<dateROUT[i]) || (currentDATE<335)){
    permRPAST=0;
    hayRLANDS=0;
    woodRLANDS=0;
    for (i=1; i<=rpasture; i++){
        if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ){
            permRPAST+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        }
    }
    CurrentSIZES=permRPAST;
}
CurrentSIZES=permRPast;
}
else if ((currentDATE>=335) && (currentDATE<=365)){
    permRPast=0;
    hayRLANDS=0;
    woodRLANDS=0;
    for (i=1; i<=rpasture; i++){
        if ( (currentDATE>=dateROUT[i]) && (currentDATE<dateRIN[i]) ){
            permRPast+=TRPAS[i];
            hayRLANDS+=HRLAND[i];
            woodRLANDS+=WRLAND[i];
        }
    }
    CurrentSIZES=permRPast;
}
else if ( (currentDATE<dateROUT[i]) || (currentDATE=dateRIN[i]) ){
    permRPast+=TRPAS[i];
    hayRLANDS+= 0;
    woodRLANDS+=WRLAND[i];
    permRPast+=HRLAND[i];
}
}
CurrentSIZES=permRPast;
}
if ((currentDATE>=1) && (currentDATE<32)){
    CarryOver=PAVAIL;
    DGRO=0;
    AGRO=0;
    HAGRO=0;
    PADJ=(SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
    HPADJ=(SOIL*SLOP*WEED*PMAIN*PREP);
    AGRO = DGRO*PADJ *.453599;
    PAVAIL = PAVAIL+(AGRO*(CurrentSIZE+CurrentSIZES));
    HAGRO= DGRO*HPADJ;
    hayAVAIL= hayAVAIL+ ( (HAGRO*hayLANDS) + (HAGRO*hayRLANDS) ) ;
    TAPG=TAPG+ ((AGRO/2.47));
    TGRO=TGRO+ (DGRO*PADJ);
if (currentDATE==31){
    JANUARYGROWTH=TGRO;
}
else if (currentDATE>=32) && (currentDATE<60)){
DGRO=0;
AGRO=0;
HAGRO=0;
PADJ=(SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
HPADJ=(SOIL*SLOP*WEED*PMAIN*PREP);
AGRO = DGRO*PADJ *.453599;
PAVAIL += (AGRO*(CurrentSIZE+CurrentSIZES));
HAGRO= DGRO*HPADJ;
hayAVAIL=(hayAVAIL+ (HAGRO*hayLANDS) + (HAGRO*hayRLANDS) )
);
TAPG=TAPG+((AGRO/2.47));
TGRO=TGRO+(DGRO*PADJ);
if (currentDATE==59){
FEBRUARYGROWTH=TGRO-(JANUARYGROWTH);
}
}
else if (currentDATE>=60) && (currentDATE<91)){
DGRO=0;
AGRO=0;
HAGRO=0;
PADJ=(SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
HPADJ=(SOIL*SLOP*WEED*PMAIN*PREP);
AGRO = DGRO*PADJ *.453599;
PAVAIL += (AGRO*(CurrentSIZE+CurrentSIZES));
HAGRO= DGRO*HPADJ;
hayAVAIL=(hayAVAIL+ (HAGRO*hayLANDS) + (HAGRO*hayRLANDS) )
);
TAPG=TAPG+((AGRO/2.47));
TGRO=TGRO+(DGRO*PADJ);
if (currentDATE==90){
   MARCHGROWTH=TGRO-(JANUARYGROWTH+FEBRUARYGROWTH);
}
else if ((currentDATE>=91) && (currentDATE<121)){
   DGRO=0;
   AGRO=0;
   HAGRO=0;
   PADJ=(SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
   HPADJ=(SOIL*SLOP*WEED*PMAIN*PREP);
   AGRO = DGRO*PADJ *.453599;
   PAVAIL += (AGRO*(CurrentSIZE+CurrentSIZES));
   HAGRO = DGRO*HPADJ;
   hayAVAIL= (hayAVAIL+ (HAGRO*hayLANDS) + (HAGRO*hayRLANDS) ) ;
   TAPG=TAPG+((AGRO/2.47));
   TGRO=TGRO+(DGRO*PADJ);
   if (currentDATE==120){
      APRILGROWTH=TGRO-(JANUARYGROWTH+FEBRUARYGROWTH+MARCHGROWTH);
   }
}
else if ((currentDATE>=121) && (currentDATE<152)){
   DGRO=0;
   AGRO=0;
   HAGRO=0;
   PADJ=(SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
   HPADJ=(SOIL*SLOP*WEED*PMAIN*PREP);
AGRO = DGRO*PADJ * .453599;
PAVAIL += (AGRO*(CurrentSIZE+CurrentSIZES));
HAGRO = DGRO*HPADJ;
hayAVAIL = (hayAVAIL + (HAGRO*hayLANDS) + (HAGRO*hayRLANDS));
TAPG = TAPG + ((AGRO/2.47));
TGRO = TGRO + (DGRO*PADJ);
if (currentDATE == 151) {
MAYGROWTH = TGRO - (JANUARYGROWTH + FEBRUARYGROWTH + MARCHGROWTH + APRILGROWTH);
}
else if ((currentDATE >= 152) && (currentDATE < 182)) {
DGRO = 0;
AGRO = 0;
HAGRO = 0;
PADJ = (SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
HPADJ = (SOIL*SLOP*WEED*PMAIN*PREP);
DGRO = PFOR1[6]*(FR[1]) + PFOR2[6]*(FR[2]) +
PFOR3[6]*(FR[3]) + PFOR4[6]*(FR[4]) +
PFOR5[6]*(FR[5]) + PFOR6[6]*(FR[6]) +
PFOR7[6]*(FR[7]) + PFOR8[6]*(FR[8]) +
PFOR9[6]*(FR[9]) + PFOR10[6]*(FR[10]) +
PFOR13[6]*(FR[13]) + PFOR14[6]*(FR[14]) +
PFOR15[6]*(FR[15]) + PFOR16[6]*(FR[16]) +
PFOR17[6]*(FR[17]) + PFOR18[6]*(FR[18]) +
PFOR19[6]*(FR[19]) + PFOR20[6]*(FR[20]) +
PFOR21[6]*(FR[21]);
AGRO = DGRO*PADJ * .453599;
PAVAIL += (AGRO*(CurrentSIZE+CurrentSIZES));
HAGRO = DGRO*HPADJ;
hayAVAIL = (hayAVAIL + (HAGRO*hayLANDS) + (HAGRO*hayRLANDS));
TAPG = TAPG + ((AGRO/2.47));
TGRO = TGRO + (DGRO*PADJ);
if (currentDATE == 181) {
JUNEGROWTH = TGRO - (JANUARYGROWTH + FEBRUARYGROWTH + MARCHGROWTH + APRILGROWTH + MAYGROWTH);
}
else if ((currentDATE >= 182) && (currentDATE < 213)) {
DGRO = 0;
AGRO = 0;
HAGRO = 0;
PADJ = (SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
HPADJ = (SOIL*SLOP*WEED*PMAIN*PREP);
DGRO = PFOR1[7]*(FR[1]) + PFOR2[7]*(FR[2]) +
PFOR3[7]*(FR[3]) + PFOR4[7]*(FR[4]) +
PFOR5[7]*(FR[5]) + PFOR6[7]*(FR[6]) +
PFOR7[7]*(FR[7]) + PFOR8[7]*(FR[8]) +
\![\text{PFOR9}[7] \times (\text{FR}[9]) + \text{PFOR10}[7] \times (\text{FR}[10]) + \\
\text{PFOR11}[7] \times (\text{FR}[11]) + \text{PFOR12}[7] \times (\text{FR}[12]) + \\
\text{PFOR13}[7] \times (\text{FR}[13]) + \text{PFOR14}[7] \times (\text{FR}[14]) + \\
\text{PFOR15}[7] \times (\text{FR}[15]) + \text{PFOR16}[7] \times (\text{FR}[16]) + \\
\text{PFOR17}[7] \times (\text{FR}[17]) + \text{PFOR18}[7] \times (\text{FR}[18]) + \\
\text{PFOR19}[7] \times (\text{FR}[19]) + \text{PFOR20}[7] \times (\text{FR}[20]) + \\
\text{PFOR21}[7] \times (\text{FR}[21])];

\text{AGRO} = \text{DGRO} \times \text{PADJ} \times 0.453599;
\text{PAVAIL} += (\text{AGRO} \times \text{CurrentSIZE} + \text{CurrentSIZES});
\text{HAGRO} = \text{DGRO} \times \text{HPADJ};
\text{hayAVAIL} = (\text{hayAVAIL} + (\text{HAGRO} \times \text{hayLANDS}) + (\text{HAGRO} \times \text{hayRLANDS}));
\text{TAPG} = \text{TAPG} + ((\text{AGRO} / 2.47));
\text{TGRO} = \text{TGRO} + (\text{DGRO} \times \text{PADJ});
\text{if} (\text{currentDATE} == 212) \{ \\
\text{JULYGROWTH} = \text{TGRO} - (\text{JANUARYGROWTH} + \text{FEBRUARYGROWTH} + \text{MARCHGROWTH} + \\
\text{APRILGROWTH} + \text{MAYGROWTH} + \text{JUNEGROWTH});
\}
\text{else if} ((\text{currentDATE} >= 213) \& \& (\text{currentDATE} < 244)) \{ \\
\text{DGRO} = 0;
\text{AGRO} = 0;
\text{HAGRO} = 0;
\text{PADJ} = (\text{SOIL} \times \text{PMAN} \times \text{SLOP} \times \text{WEED} \times \text{PMAIN} \times \text{PREP});
\text{HPADJ} = (\text{SOIL} \times \text{SLOP} \times \text{WEED} \times \text{PMAIN} \times \text{PREP});
\text{DGRO} = \text{PFOR1}[8] \times (\text{FR}[1]) + \text{PFOR2}[8] \times (\text{FR}[2]) + \\
\text{PFOR3}[8] \times (\text{FR}[3]) + \text{PFOR4}[8] \times (\text{FR}[4]) + \\
\text{PFOR5}[8] \times (\text{FR}[5]) + \text{PFOR6}[8] \times (\text{FR}[6]) + \\
\text{PFOR7}[8] \times (\text{FR}[7]) + \text{PFOR8}[8] \times (\text{FR}[8]) + \\
\text{PFOR9}[8] \times (\text{FR}[9]) + \text{PFOR10}[8] \times (\text{FR}[10]) + \\
\text{PFOR11}[8] \times (\text{FR}[11]) + \text{PFOR12}[8] \times (\text{FR}[12]) + \\
\text{PFOR13}[8] \times (\text{FR}[13]) + \text{PFOR14}[8] \times (\text{FR}[14]) + \\
\text{PFOR15}[8] \times (\text{FR}[15]) + \text{PFOR16}[8] \times (\text{FR}[16]) + \\
\text{PFOR17}[8] \times (\text{FR}[17]) + \text{PFOR18}[8] \times (\text{FR}[18]) + \\
\text{PFOR19}[8] \times (\text{FR}[19]) + \text{PFOR20}[8] \times (\text{FR}[20]) + \\
\text{PFOR21}[8] \times (\text{FR}[21]);
\text{AGRO} = \text{DGRO} \times \text{PADJ} \times 0.453599;
\text{PAVAIL} += (\text{AGRO} \times \text{CurrentSIZE} + \text{CurrentSIZES});
\text{HAGRO} = \text{DGRO} \times \text{HPADJ};
\text{hayAVAIL} = (\text{hayAVAIL} + (\text{HAGRO} \times \text{hayLANDS}) + (\text{HAGRO} \times \text{hayRLANDS}));
\text{TAPG} = \text{TAPG} + ((\text{AGRO} / 2.47));
\text{TGRO} = \text{TGRO} + (\text{DGRO} \times \text{PADJ});
\text{if} (\text{currentDATE} == 243) \{ \\
\text{AUGUSTGROWTH} = \text{TGRO} - (\text{JANUARYGROWTH} + \text{FEBRUARYGROWTH} + \text{MARCHGROWTH} + \\
\text{APRILGROWTH} + \text{MAYGROWTH} + \text{JUNEGROWTH} + \text{JULYGROWTH});
\}
\text{else if} ((\text{currentDATE} >= 244) \& \& (\text{currentDATE} < 274)) \{ \\
\text{DGRO} = 0;
\text{AGRO} = 0;
HAGRO=0;
PADJ=(SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
HPADJ=(SOIL*SLOP*WEED*PMAIN*PREP);
DGRO=PFOR1[9]*(FR[1])+PFOR2[9]*(FR[2])+
PFOR3[9]*(FR[3])+PFOR4[9]*(FR[4])+
PFOR5[9]*(FR[5])+PFOR6[9]*(FR[6])+
PFOR7[9]*(FR[7])+PFOR8[9]*(FR[8])+
PFOR9[9]*(FR[9])+PFOR10[9]*(FR[10])+
PFOR13[9]*(FR[13])+PFOR14[9]*(FR[14])+
PFOR15[9]*(FR[15])+PFOR16[9]*(FR[16])+
PFOR17[9]*(FR[17])+PFOR18[9]*(FR[18])+
PFOR19[9]*(FR[19])+PFOR20[9]*(FR[20])+
PFOR21[9]*(FR[21]);
AGRO = DGRO*PADJ *.453599;
PAVAIL += (AGRO*(Current SIZE+Current SIZES));
HAGRO= DGRO*HPADJ;
hayAVAIL=(hayAVAIL+( (HAGRO*hayLANDS) + (HAGRO*hayRLANDS) ) )
);
TAPG=TAPG+( (AGRO/2.47) )
TGRO=TGRO+ (DGRO*PADJ);
if (currentDATE==273) {
   SEPTEMBERGROWTH=TGRO-
   (JANUARYGROWTH+FEBRUARYGROWTH+MARCHGROWTH+
   APRILGROWTH+MAYGROWTH+JUNEGROWTH+JULYGROWTH+
   AUGUSTGROWTH);
}
else if ((currentDATE>=274) && (currentDATE<305)) {
   DGRO=0;
   AGRO=0;
   HAGRO=0;
   PADJ=(SOIL*PMAN*SLOP*WEED*PMAIN*PREP);
   HPADJ=(SOIL*SLOP*WEED*PMAIN*PREP);
   DGRO=PFOR1[10]*(FR[1])+PFOR2[10]*(FR[2])+
   PFO3[10]*(FR[3])+PFOR4[10]*(FR[4])+
   PFOR5[10]*(FR[5])+PFOR6[10]*(FR[6])+
   PFOR7[10]*(FR[7])+PFOR8[10]*(FR[8])+
   PFOR9[10]*(FR[9])+PFOR10[10]*(FR[10])+
   PFOR13[10]*(FR[13])+PFOR14[10]*(FR[14])+
   PFOR15[10]*(FR[15])+PFOR16[10]*(FR[16])+
   PFOR17[10]*(FR[17])+PFOR18[10]*(FR[18])+
   PFOR19[10]*(FR[19])+PFOR20[10]*(FR[20])+
   PFOR21[10]*(FR[21]);
   AGRO = DGRO*PADJ *.453599;
   PAVAIL += (AGRO*(Current SIZE+Current SIZES));
   HAGRO= DGRO*HPADJ;
   hayAVAIL=(hayAVAIL+( (HAGRO*hayLANDS) + (HAGRO*hayRLANDS) ) )
);
   TAPG=TAPG+( (AGRO/2.47) )
}
TGRO = TGRO + (DGRO * PADJ);
if (currentDATE == 304) {
    OCTOBERGROWTH = TGRO -
        (JANUARYGROWTH + FEBRUARYGROWTH + MARCHGROWTH +
        APRILGROWTH + MAYGROWTH + JUNEGROWTH + JULYGROWTH +
        AUGUSTGROWTH + SEPTEMBERGROWTH);
} } else if ((currentDATE >= 305) && (currentDATE < 335)) {
    DGRO = 0;
    AGRO = 0;
    HAGRO = 0;
    PADJ = (SOIL * PMAN * SLOP * WEED * PMAIN * PREP);
    HPADJ = (SOIL * SLOP * WEED * PMAIN * PREP);
    AGRO = DGRO * PADJ * .453599;
    PAVAIL += (AGRO * (CurrentSIZE + CurrentSIZES));
    HAGRO = DGRO * HPADJ;
    hayAVAIL = (hayAVAIL + (HAGRO * hayLANDS) + (HAGRO * hayRLANDS));
    TAPG = TAPG + ((AGRO / 2.47));
    TGRO = TGRO + (DGRO * PADJ);
} if (currentDATE == 334) {
    NOVEMBERGROWTH = TGRO -
        (JANUARYGROWTH + FEBRUARYGROWTH + MARCHGROWTH +
        APRILGROWTH + MAYGROWTH + JUNEGROWTH + JULYGROWTH +
        AUGUSTGROWTH + SEPTEMBERGROWTH + OCTOBERGROWTH);
} } else if ((currentDATE >= 335) && (currentDATE <= 365)) {
    DGRO = 0;
    AGRO = 0;
    HAGRO = 0;
    PADJ = (SOIL * PMAN * SLOP * WEED * PMAIN * PREP);
    HPADJ = (SOIL * SLOP * WEED * PMAIN * PREP);
        PFOR21[12] * (FR[21]);
    AGRO = DGRO * PADJ * .453599;
    PAVAIL += (AGRO * (CurrentSIZE + CurrentSIZES));
    HAGRO = DGRO * HPADJ;
    hayAVAIL = (hayAVAIL + (HAGRO * hayLANDS) + (HAGRO * hayRLANDS));
    TAPG = TAPG + ((AGRO / 2.47));
    TGRO = TGRO + (DGRO * PADJ);
}
PFOR13[12]*(FR[13]) + PFORe14[12]*(FR[14]) +
PFOR15[12]*(FR[15]) + PFORe16[12]*(FR[16]) +
PFOR17[12]*(FR[17]) + PFORe18[12]*(FR[18]) +
PFOR19[12]*(FR[19]) + PFORe20[12]*(FR[20]) +
PFOR21[12]*(FR[21]);
AGRO = DGRO*PADJ *.453599;
PAVAIL += (AGRO*(CurrentSIZE+CurrentSIZES));
HAGRO = DGRO*HPADJ;
hayAVAIL=(hayAVAIL+ (HAGRO*hayLANDS) + (HAGRO*hayRLANDS) ) ;
TAPG=TAPG+((AGRO/2.47));
TGRO=TGRO+(DGRO*PADJ);
if (currentDATE==365){
DECEMBREGROWTH=TGRO-
(JANUARYGROWTH+FEBRUARYGROWTH+MARCHGROWTH+
APRILGROWTH+MAYGROWTH+JUNEGROWTH+JULYGROWTH+
AUGUSTGROWTH+SEPTEMBERGROWTH+OCTOBERGROWTH+
NOVEMBERGROWTH);
TTGRO=TGRO*(CurrentSIZE + CurrentSIZES) - hayAVAIL;
}

void Forage::ResetForageVariables(){
CarryOver=0;
APRILGROWTH=0;
AUGUSTGROWTH=0;
DECEMBREGROWTH=0;
FEBRUARYGROWTH=0;
JANUARYGROWTH=0;
JULYGROWTH=0;
JUNEGROWTH=0;
MARCHGROWTH=0;
MAYGROWTH=0;
NOVEMBERGROWTH=0;
OCTOBERGROWTH=0;
RAINAPRIL=0;
RAINAUGUST=0;
RAINDECEMBER=0;
RAINFEBRUARY=0;
RAINJANUARY=0;
RAINJULY=0;
RAINJUNE=0;
RAINMARCH=0;
RAINMAY=0;
RAINNOVEMBER=0;
RAINOCTOBER=0;
RAINSEPTEMBER=0;
SEPTEMBERGROWTH=0;
TAPG=0;
TGRO=0;
hayAVAIL=0;
TotAnnSen=0;
}

void Forage::ResetForageConCntr()
{
FAME=0;
FANEM=0;
FANEG=0;
TDNP=0;
mainFDM=0;
}
HerdSim.cpp
/* HerdSim.cpp : Defines the class
behaviors for the application.*/
#include "stdafx.h"
#include "HerdSim.h"
#include "HerdSimDlg.h"
#include "main.h"
#include <afx.h>
 ifndef _DEBUG
 define new DEBUG_NEW
 undef THIS_FILE
 static char THIS_FILE[] = __FILE__;
 endif
 BEGIN_MESSAGE_MAP(CHerdSimApp, CWinApp)
 ON_COMMAND(ID_HELP_CONTENTS, OnHelpContents)
 ON_COMMAND(ID_HELP, CWinApp::OnHelp)
 END_MESSAGE_MAP()

 // CHerdSimApp construction
 CHerdSimApp theApp;

 BOOL CHerdSimApp::InitInstance()
 {
 #ifdef _AFXDLL
 Enable3dControls();
 #else
 Enable3dControlsStatic();
 #endif
 CHerdSimDlg dlg;
 int nResponse = dlg.DoModal();
 if (nResponse == IDOK)
 {
 TRACE("Tried to start main\n");
 Main* main = new Main;
 main->Start();
 delete main;
 }
 else if (nResponse == IDCANCEL)
 {
 }
 return FALSE;
 }

 void CHerdSimApp::OnHelpContents()
WinHelp(0,HELP_CONTENTS);
}
HerdSimDlg.cpp
#include "stdafx.h"
#include "HerdSim.h"
#include "HerdSimDlg.h"
#include <afx.h>
#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
CAboutDlg();
enum { IDD = IDD_ABOUTBOX };  
DECLARE_MESSAGE_MAP();
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// CHerdSimDlg dialog
CHerdSimDlg::CHerdSimDlg(CWnd* pParent /*=NULL*/) : CDialog(CHerdSimDlg::IDD, pParent)
{
m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CHerdSimDlg::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CHerdSimDlg, CDialog)
ON_WM_SYSCOMMAND()
ON_WM_PAINT()
ON_WM_QUERYDRAGICON()
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()
BOOL CHerdSimDlg::OnInitDialog()  
{  
CDialog::OnInitDialog();  
ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);  
ASSERT(IDM_ABOUTBOX < 0xF000);  
CMenu* pSysMenu = GetSystemMenu(FALSE);  
CString strAboutMenu;  
strAboutMenu.LoadString(IDS_ABOUTBOX);  
if (!strAboutMenu.IsEmpty())  
{  
pSysMenu->AppendMenu(MF_SEPARATOR);  
pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);  
}  
setIcon(m_hIcon, TRUE);  
setIcon(m_hIcon, FALSE);  
return TRUE;  
}  

void CHerdSimDlg::OnSysCommand(UINT nID, LPARAM lParam)  
{  
if ((nID & 0xFFF0) == IDM_ABOUTBOX)  
{  
CAboutDlg dlgAbout;  
dlgAbout.DoModal();  
}  
else  
{  
CDialog::OnSysCommand(nID, lParam);  
}  
}  

void CHerdSimDlg::OnPaint()  
{  
if (IsIconic())  
{  
CPaintDC dc(this);  
SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);  
int cxIcon = GetSystemMetrics(SM_CXICON);  
int cyIcon = GetSystemMetrics(SM_CYICON);  
CRect rect;  
GetClientRect(&rect);  
int x = (rect.Width() - cxIcon + 1) / 2;  
int y = (rect.Height() - cyIcon + 1) / 2;  
dc.DrawIcon(x, y, m_hIcon);  
}  
else  
{  
CDialog::OnPaint();  
}  
}
HCURSOR CHerdSimDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

// CInput2 dialog
CInput2::CInput2(CWnd* pParent /*=NULL*/)
    : CDialoque(CInput2::IDD, pParent)
{
    startrefresh=1;
    //{{AFX_DATA_INIT(CInput2)
    m_AI = FALSE;
    m_black = FALSE;
    m_buypreg = FALSE;
    m_conum = 0;
    m_costbull = 0.0;
    m_cullpol = -1;
    m_implantheifer = FALSE;
    m_implantsheifer = FALSE;
    m_implantssteer = FALSE;
    m_implantsteer = FALSE;
    m_iono = FALSE;
    m_numbulls = 0;
    m_salvbull = 0.0;
    m_yearsbull = 0;
    m_shimplants = 0;
    m_ssimplants = 0;
    m_cullage = -1;
    m_buycalves = FALSE;
    m_selecwiherd = FALSE;
    m_repbreed = -1;
    m_reppure = -1;
    }

    void CInput2::DoDataExchange(CDataExchange* pDX)
    {
        CDialog::DoDataExchange(pDX);
        DDX_Control(pDX, IDC_NEXT, m_next);
        DDX_Check(pDX, IDC_AI, m_AI);
        DDX_Check(pDX, IDC_BLACK, m_black);
        DDX_Check(pDX, IDC_BUYPREG, m_buypreg);
        DDX_Text(pDX, IDC_CONUM, m_conum);
        DDV_MinMaxInt(pDX, m_conum, 1, 9999);
        DDX_Text(pDX, IDC_COSTBULL, m_costbull);
        DDV_MinMaxDouble(pDX, m_costbull, 0., 9999.99);
        DDX_CBIndex(pDX, IDC_CULLPOL, m_cullpol);
        DDX_Check(pDX, IDC_IMPLANTHEIFER, m_implantheifer);
        DDX_Check(pDX, IDC_IMPLANTSHEIFER, m_implantsheifer);
        DDX_Check(pDX, IDC_IMPLANTSTEER, m_implantsteer);
        DDX_Check(pDX, IDC_REPBREED, m_repbreed);
        DDX_Check(pDX, IDC_REPPURE, m_reppure);
    }
DDX_Check(pDX, IDC_IMPLANTSSTEER, m_implantssteer);
DDX_Check(pDX, IDC_IMPLANTSTEER, m_implantsteer);
DDX_Check(pDX, IDC_IONO, m_iono);
DDX_Text(pDX, IDC_NUMBULLS, m_numbulls);
DDV_MinMaxInt(pDX, m_numbulls, 0, 9999);
DDX_Text(pDX, IDC_SALVBULL, m_salvbull);
DDV_MinMaxDouble(pDX, m_salvbull, 0., 1.);
DDX_Text(pDX, IDC_YEARSBULL, m_yearsbull);
DDV_MinMaxInt(pDX, m_yearsbull, 0, 999);
DDX_Text(pDX, IDC_SHEDIT, m_shimplants);
DDV_MinMaxInt(pDX, m_shimplants, 0, 5);
DDX_Text(pDX, IDC_SSEDIT, m_ssimplants);
DDV_MinMaxInt(pDX, m_ssimplants, 0, 5);
DDX_CBIndex(pDX, IDC_CULLAGE, m_cullage);
DDX_Check(pDX, IDC_BUYCALVES, m_buycalves);
DDX_Check(pDX, IDC_SELECWIHERD, m_selecwiherd);
DDX_CBIndex(pDX, IDC_REPBREED, m_repbreed);
DDX_Radio(pDX, IDC_REPPURE, m_reppure);
BEGIN_MESSAGE_MAP(CInput2, CDialog)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_BN_CLICKED(IDC_IMPLANTSSTEER, OnImplantssteer)
ON_BN_CLICKED(IDC_IMPLANTSTEER, OnImplantsteer)
ON_BN_CLICKED(IDC_IONO, m_iono);
ON_BN_CLICKED(IDC_NUMBULLS, m_numbulls);
ON_BN_CLICKED(IDC_SALVBULL, m_salvbull);
ON_BN_CLICKED(IDC_YEARSBULL, m_yearsbull);
ON_BN_CLICKED(IDC_SHEDIT, m_shimplants);
ON_BN_CLICKED(IDC_SSEDIT, m_ssimplants);
ON_BN_CLICKED(IDC_CULLAGE, m_cullage);
ON_BN_CLICKED(IDC_BUYCALVES, m_buycalves);
ON_BN_CLICKED(IDC_SELECWIHERD, m_selecwiherd);
ON_BN_CLICKED(IDC_REPBREED, m_repbreed);
ON_BN_CLICKED(IDC_REPPURE, m_reppure);
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CInput1, CDialog)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_BN_CLICKED(IDC_IMPLANTSSTEER, OnImplantssteer)
ON_BN_CLICKED(IDC_IMPLANTSTEER, OnImplantsteer)
ON_BN_CLICKED(IDC_IONO, m_iono);
ON_BN_CLICKED(IDC_NUMBULLS, m_numbulls);
ON_BN_CLICKED(IDC_SALVBULL, m_salvbull);
ON_BN_CLICKED(IDC_YEARSBULL, m_yearsbull);
ON_BN_CLICKED(IDC_SHEDIT, m_shimplants);
ON_BN_CLICKED(IDC_SSEDIT, m_ssimplants);
ON_BN_CLICKED(IDC_CULLAGE, m_cullage);
ON_BN_CLICKED(IDC_BUYCALVES, m_buycalves);
ON_BN_CLICKED(IDC_SELECWIHERD, m_selecwiherd);
ON_BN_CLICKED(IDC_REPBREED, m_repbreed);
ON_BN_CLICKED(IDC_REPPURE, m_reppure);
END_MESSAGE_MAP()

// CInput2 message handlers
// CInput1 dialog
CInput1::CInput1(CWnd* pParent /*=NULL*/)
: CDialog(CInput1::IDD, pParent)
{
    m_address = _T("");
    m_farman = _T("");
    m_farmname = _T("");
    m_farman = _T("");
    m_liveman = _T("");
}

void CInput1::DoDataExchange(CDataExchange* pDX)
{
    CDIalog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CInput1)
}
DDX_Control(pDX, IDC_NEXT, m_next);
DDX_Text(pDX, IDC_ADDRESS, m_address);
DDV_MaxChars(pDX, m_address, 40);
DDX_Text(pDX, IDC_FARMMAN, m_farmman);
DDV_MaxChars(pDX, m_farmman, 40);
DDX_Text(pDX, IDC_FARMNAME, m_farmname);
DDV_MaxChars(pDX, m_farmname, 40);
DDX_Text(pDX, IDC_FARMOWNER, m_farmowner);
DDV_MaxChars(pDX, m_farmowner, 40);
DDX_Text(pDX, IDC_LIVEMAN, m_liveman);
DDV_MaxChars(pDX, m_liveman, 40);
}

BEGIN_MESSAGE_MAP(CInput1, CDialog)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_BN_CLICKED(IDC_MENU, OnMenu)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CMainMenu, CDialog)
ON_BN_CLICKED(IDC_COW, OnCow)
ON_BN_CLICKED(IDC_FORAGE, OnForage)
ON_BN_CLICKED(IDC_MARKET, OnMarket)
ON_BN_CLICKED(IDC_NUTRIT, OnNutrit)
ON_BN_CLICKED(IDC_QUITSIM, OnQuitsim)
ON_BN_CLICKED(IDC_STARTSIM, OnStartsim)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

// CInput1 message handlers

void CMainMenu::DoDataExchange(CDataExchange* pDX)
{
    CDiagol::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_STARTSIM, m_startsim);
    DDX_Control(pDX, IDC_QUITSIM, m_quitsim);
    DDX_Control(pDX, IDC_NUTRIT, m_nutrit);
    DDX_Control(pDX, IDC_MARKET, m_market);
    DDX_Control(pDX, IDC_FORAGE, m_forage);
    DDX_Control(pDX, IDC_COW, m_cow);
}

BEGIN_MESSAGE_MAP(CMainMenu, CDialog)
ON_BN_CLICKED(IDC_COW, OnCow)
ON_BN_CLICKED(IDC_FORAGE, OnForage)
ON_BN_CLICKED(IDC_MARKET, OnMarket)
ON_BN_CLICKED(IDC_NUTRIT, OnNutrit)
ON_BN_CLICKED(IDC_QUITSIM, OnQuitsim)
ON_BN_CLICKED(IDC_STARTSIM, OnStartsim)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()
// CInput2A dialog
CInput2A::CInput2A(CWnd* pParent /*=NULL*/)
: CDialog(CInput2A::IDD, pParent)
{
    m_angus = 0.0;
    m_hereford = 0.0;
    m_phereford = 0.0;
    m_shorthorn = 0.0;
    m_charol = 0.0;
    m_chianina = 0.0;
    m_bred = -1;
    m_bullbreed = -1;
    m_gelb = 0.0;
    m_limousin = 0.0;
    m_maine = 0.0;
    m_pinz = 0.0;
    m_pure = -1;
    m_simmental = 0.0;
    m_tarentaise = 0.0;
    m_ezcalf = FALSE;
    m_ezbullbreed = -1;
    m_calfease = FALSE;
}

void CInput2A::DoDataExchange(CDataExchange* pDX)
{
    CDlgDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_ANGUS, m_angus);
    DDV_MinMaxDouble(pDX, m_angus, 0., 100.);
    DDX_Text(pDX, IDC_HEREFORD, m_hereford);
    DDV_MinMaxDouble(pDX, m_hereford, 0., 100.);
    DDX_MinMaxDouble(pDX, m_phereford, 0., 100.);
    DDX_Text(pDX, IDC_SHORTHORN, m_shorthorn);
    DDV_MinMaxDouble(pDX, m_shorthorn, 0., 100.);
    DDX_Text(pDX, IDC_CHAROL, m_charol);
    DDV_MinMaxDouble(pDX, m_charol, 0., 100.);
    DDX_Text(pDX, IDC_CHIANINA, m_chianina);
    DDV_MinMaxDouble(pDX, m_chianina, 0., 100.);
    DDX_Radio(pDX, IDC_BRED, m_bred);
    DDX_CBIndex(pDX, IDC_BULLBREED, m_bullbreed);
    DDX_Text(pDX, IDC_SHORTHORN, m_shorthorn);
    DDX_MinMaxDouble(pDX, m_shorthorn, 0., 100.);
    DDX_Text(pDX, IDC_CHAROL, m_charol);
    DDX_MinMaxDouble(pDX, m_charol, 0., 100.);
    DDX_Text(pDX, IDC_CHIANINA, m_chianina);
    DDX_MinMaxDouble(pDX, m_chianina, 0., 100.);
    DDX_Radio(pDX, IDC_BRED, m_bred);
    DDX_CBIndex(pDX, IDC_BULLBREED, m_bullbreed);
    DDX_Text(pDX, IDC_GELB, m_gelb);
    DDV_MinMaxDouble(pDX, m_gelb, 0., 100.);
    DDX_Text(pDX, IDC_LIMOUSIN, m_limousin);
    DDV_MinMaxDouble(pDX, m_limousin, 0., 100.);
    DDX_Text(pDX, IDC_MAINE, m_maine);
    DDV_MinMaxDouble(pDX, m_maine, 0., 100.);
    DDX_Text(pDX, IDC_PINZ, m_pinz);
    DDV_MinMaxDouble(pDX, m_pinz, 0., 100.);
    DDX_CBIndex(pDX, IDC_PURE, m_pure);
    DDX_Text(pDX, IDC_SIMMENTAL, m_simmental);
DDV_MinMaxDouble(pDX, m_simmental, 0., 100.);
DDX_Text(pDX, IDC_TARENTAISE, m_tarentaise);
DDV_MinMaxDouble(pDX, m_tarentaise, 0., 100.);
DDX_Check(pDX, IDC_EZCALF, m_ezcalf);
DDX_CBIndex(pDX, IDC_EZBULLBREED, m_ezbullbreed);
DDX_Check(pDX, IDC_CALFEASE, m_calfease);
}
BEGIN_MESSAGE_MAP(CInput2A, CDialog)
ON_BN_CLICKED(IDC_BRED, OnBred)
ON_BN_CLICKED(IDC_EZCALF, OnEzCalf)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MENU, OnMenu)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_BN_CLICKED(IDC_CROSSBRED, OnBred)
ON_BN_CLICKED(IDC_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

// CInput2A message handlers

CInput3::CInput3(CWnd* pParent /*=NULL*/)
: CDialog(CInput3::IDD, pParent)
{
    startrefresh=1;
    m_culled = -1;
    m_diff = 0.0;
    m_heifer = -1;
    m_sheifer = -1;
    m_ssteer = -1;
    m_steer = -1;
    m_year1 = 0.00;
    m_year2 = 0.00;
    m_year3 = 0.00;
    m_orphan = -1;
    m_hedg1 = FALSE;
    m_hedg2 = FALSE;
    m_onite = FALSE;
    m_dist = 0;
}

void CInput3::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_NEXT, m_next);
    DDX_Radio(pDX, IDC_CULLED, m_culled);
    DDX_Text(pDX, IDC_DIFF, m_diff);
    DDX_MinMaxDouble(pDX, m_diff, -9999., 9999.);
    DDX_Check(pDX, IDC_HEDG1, m_hedg1);
    DDX_Check(pDX, IDC_HEDG2, m_hedg2);
    DDX_Radio(pDX, IDC_HEIFER, m_heifer);
    DDX_Radio(pDX, IDC_SHEIFER, m_sheifer);
    DDX_Radio(pDX, IDC_SSTEER, m_ssteer);
}
DDX_Radio(pDX, IDC_STEER, m_steer);
DDX_Text(pDX, IDC_YEAR1, m_year1);
DDV_MinMaxDouble(pDX, m_year1, 0., 9999.);
DDX_Text(pDX, IDC_YEAR2, m_year2);
DDV_MinMaxDouble(pDX, m_year2, 0., 9999.);
DDX_Text(pDX, IDC_YEAR3, m_year3);
DDV_MinMaxDouble(pDX, m_year3, 0., 9999.);
DDX_Text(pDX, IDC_SYEAR1, m_syear1);
DDV_MinMaxDouble(pDX, m_syear1, 0., 9999.);
DDX_Text(pDX, IDC_SYEAR2, m_syear2);
DDV_MinMaxDouble(pDX, m_syear2, 0., 9999.);
DDX_Text(pDX, IDC_SYEAR3, m_syear3);
DDV_MinMaxDouble(pDX, m_syear3, 0., 9999.);
DDX_Text(pDX, IDC_CYEAR1, m_cyear1);
DDV_MinMaxDouble(pDX, m_cyear1, 0., 9999.);
DDX_Text(pDX, IDC_CYEAR2, m_cyear2);
DDV_MinMaxDouble(pDX, m_cyear2, 0., 9999.);
DDX_Text(pDX, IDC_CYEAR3, m_cyear3);
DDV_MinMaxDouble(pDX, m_cyear3, 0., 9999.);
DDX_Radio(pDX, IDC_ORPHAN, m_orphan);
DDX_Text(pDX, IDC_DIST, m_dist);
DDV_MinMaxDouble(pDX, m_dist, -9999., 9999.);
DDX_Check(pDX, IDC_ONITE, m_onite);
}
BEGIN_MESSAGE_MAP(CInput3, CDialog)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_BN_CLICKED(IDC_MENU, OnMenu)
ON_BN_CLICKED(IDC_HEDG1, OnHedgecalves)
ON_BN_CLICKED(IDC_HEDG2, OnHedgecalves)
ON_BN_CLICKED(IDC_RADIO2, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO3, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO4, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO5, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO7, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO8, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO10, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO11, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO13, OnMarketingcattle)
ON_BN_CLICKED(IDC_RADIO14, OnMarketingcattle)
ON_BN_CLICKED(IDC_ORPHAN, OnMarketingcattle)
ON_BN_CLICKED(IDC_CULLED, OnMarketingcattle)
ON_BN_CLICKED(IDC_STEER, OnMarketingcattle)
ON_BN_CLICKED(IDC_HEIFER, OnMarketingcattle)
ON_BN_CLICKED(IDC_SSTEER, OnMarketingcattle)
ON_BN_CLICKED(IDC_SHEIFER, OnMarketingcattle)
END_MESSAGE_MAP()
///////////////////////////////////////////
// CInput3 message handlers
///////////////////////////////////////////
CInput4::CInput4(CWnd* pParent /*=NULL*/)
: CDiallog(CInput4::IDD, pParent)
{
    m_aicharge = 0.0;
    m_calftrans = 0.0;
    m_fence = 0.0;
    m_implants = 0.0;
    m_insurance = 0.0;
    m_intr = 0.0;
    m_intrs = 0.0;
    m_labor = 0.0;
    m_laborcharge = 0.0;
    m_machine = 0.0;
    m_maint = 0.0;
    m_mincosts = 0.0;
    m_sfence = 0.0;
    m_slabor = 0.0;
    m_smachine = 0.0;
    m_smaint = 0.0;
    m_smincost = 0.0;
    m_ssupplies = 0.0;
    m_strans = 0.0;
    m_supplies = 0.0;
    m_sutils = 0.0;
    m_svet = 0.0;
    m_taxrate = 0.0;
    m_trans = 0.0;
    m_utils = 0.0;
    m_vet = 0.0;
}

void CInput4::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_AICHARGE, m_aicharge);
    DDV_MinMaxDouble(pDX, m_aicharge, 0., 9999.);
    DDX_Text(pDX, IDC_CALFTRANS, m_calftrans);
    DDV_MinMaxDouble(pDX, m_calftrans, 0., 99999.);
    DDX_Text(pDX, IDC_FENCE, m_fence);
    DDV_MinMaxDouble(pDX, m_fence, 0., 9999.);
    DDX_Text(pDX, IDC_IMPLANTS, m_implants);
    DDV_MinMaxDouble(pDX, m_implants, 0., 9999.);
    DDX_Text(pDX, IDC_INSURANCE, m_insurance);
    DDV_MinMaxDouble(pDX, m_insurance, 0., 9999.);
    DDX_Text(pDX, IDC_LABOR, m_labor);
    DDV_MinMaxDouble(pDX, m_labor, 0., 9999.);
    DDX_Text(pDX, IDC_LABORCHARGE, m_laborcharge);
    DDV_MinMaxDouble(pDX, m_laborcharge, 0., 9999.);
    DDX_Text(pDX, IDC_INTR, m_intr);
    DDV_MinMaxDouble(pDX, m_intr, 0., 9999.);
    DDX_Text(pDX, IDC_INTRS, m_intrs);
    DDV_MinMaxDouble(pDX, m_intrs, 0., 9999.);
DDX_Text(pDX, IDC_MACHINE, m_machine);
DDV_MinMaxDouble(pDX, pDX, m_machine, 0., 999999.);
DDX_Text(pDX, IDC_MAINT, m_maint);
DDV_MinMaxDouble(pDX, pDX, m_maint, 0., 9999.);
DDX_Text(pDX, IDC_MINCOSTS, m_mincosts);
DDV_MinMaxDouble(pDX, pDX, m_mincosts, 0., 99999.);
DDX_Text(pDX, IDC_SFENCE, m_sfence);
DDV_MinMaxDouble(pDX, pDX, m_sfence, 0., 99999.);
DDX_Text(pDX, IDC_SLABOR, m_slabor);
DDV_MinMaxDouble(pDX, pDX, m_slabor, 0., 99999.);
DDX_Text(pDX, IDC_SMACHINE, m_smachine);
DDV_MinMaxDouble(pDX, pDX, m_smachine, 0., 999999.);
DDX_Text(pDX, IDC_SMAINT, m_smaint);
DDV_MinMaxDouble(pDX, pDX, m_smaint, 0., 99999.);
DDX_Text(pDX, IDC_SMINCOST, m_smincost);
DDV_MinMaxDouble(pDX, pDX, m_smincost, 0., 999999.);
DDX_Text(pDX, IDC_SSUPPLIES, m_ssupplies);
DDV_MinMaxDouble(pDX, pDX, m_ssupplies, 0., 99999.);
DDX_Text(pDX, IDC_STRANS, m_strans);
DDV_MinMaxDouble(pDX, pDX, m_strans, 0., 99999.);
DDX_Text(pDX, IDC_SUPPLIES, m_supplies);
DDV_MinMaxDouble(pDX, pDX, m_supplies, 0., 99999.);
DDX_Text(pDX, IDC_SUTILS, m_sutils);
DDV_MinMaxDouble(pDX, pDX, m_sutils, 0., 99999.);
DDX_Text(pDX, IDC_SVET, m_svet);
DDV_MinMaxDouble(pDX, pDX, m_svet, 0., 99999.);
DDX_Text(pDX, IDC_TAXRATE, m_taxrate);
DDV_MinMaxDouble(pDX, pDX, m_taxrate, 0., 1.);
DDX_Text(pDX, IDC_TRANS, m_trans);
DDV_MinMaxDouble(pDX, pDX, m_trans, 0., 99999.);
DDX_Text(pDX, IDC_UTILS, m_utils);
DDV_MinMaxDouble(pDX, pDX, m_utils, 0., 99999.);
DDX_Text(pDX, IDC_VET, m_vet);
DDV_MinMaxDouble(pDX, pDX, m_vet, 0., 99999.);
}

BEGIN_MESSAGE_MAP(CInput4, CDialog)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MENU, OnMenu)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

/// //////////////////////////////////////////////////////////////////////
// CInput4 message handlers
/// //////////////////////////////////////////////////////////////////////
CInput5::CInput5(CWnd* pParent /*=NULL*/) : CDialog(CInput5::IDD, pParent)
{
    m_erosion = -1;
    m_graze = -1;
    m_maint = -1;
}
m_region = -1;
m_rentalrate = 0.0;
m_respadd = 0.0;
m_respmaint = 0.0;
m_slope = 0.0;
mTerrain = -1;
m_weed = -1;
m_sellhay=0.0;
m_haycost=0.0;
}
void CInput5::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Radio(pDX, IDC_EROSION, m_erosion);
    DDX_Radio(pDX, IDC_GRAZE, m_graze);
    DDX_Radio(pDX, IDC_MAINT, m_maint);
    DDX_Radio(pDX, IDC_REGION, m_region);
    DDX_Text(pDX, IDC_RENTALRATE, m_rentalrate);
    DDV_MinMaxDouble(pDX, m_rentalrate, 0., 99999.);
    DDX_Text(pDX, IDC_RESPADD, m_respadd);
    DDV_MinMaxDouble(pDX, m_respadd, 0., 100.);
    DDX_Text(pDX, IDC_RESPMAINT, m_respmaint);
    DDV_MinMaxDouble(pDX, m_respmaint, 0., 100.);
    DDX_Text(pDX, IDC_SLOPE, m_slope);
    DDV_MinMaxDouble(pDX, m_slope, 0., 90.);
    DDX_Radio(pDX, IDC_TERRAIN, m_terrain);
    DDX_Radio(pDX, IDC_WEED, m_weed);
    DDX_Text(pDX, IDC_SELLHAY, m_sellhay);
    DDV_MinMaxDouble(pDX, m_sellhay, 0., 99999.);
    DDX_Text(pDX, IDC_HAYCOST, m_haycost);
    DDV_MinMaxDouble(pDX, m_haycost, 0., 99999.);
}
BEGIN_MESSAGE_MAP(CInput5, CDialog)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_BN_CLICKED(IDC_MENU, OnMenu)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////////////////
// CInput5 message handlers
////////////////////////////////////////////////////////////////////////////////////////
CInput6::CInput6(CWnd* pParent /*=NULL*/)
    : CDialog(CInput6::IDD, pParent)
{
    m_alfalfa = 0.0;
    m_bahia = 0.0;
    m_bermuda100 = 0.0;
    m_bermuda200 = 0.0;
    m_bermuda250 = 0.0;
}
m_bermudar = 0.0;
m_dallis = 0.0;
m_fescue = 0.0;
m_fescuelad = 0.0;
m_kentucky = 0.0;
m_kentuckyw = 0.0;
m_kudza = 0.0;
m_lespedza = 0.0;
m_orchard = 0.0;
m_orchardl = 0.0;
m_orchardr = 0.0;
m_pearl = 0.0;
m_rye = 0.0;
m_sericea = 0.0;
m_sudan = 0.0;
m_switch = 0.0;
m_total = 0.0;

void CInput6::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_ALFALFA, m_alfalfa);
    DDV_MinMaxDouble(pDX, m_alfalfa, 0., 1.);
    DDX_Text(pDX, IDC_BAHIA, m_bahia);
    DDV_MinMaxDouble(pDX, m_bahia, 0., 1.);
    DDX_Text(pDX, IDC_BERMUDA100, m_bermuda100);
    DDV_MinMaxDouble(pDX, m_bermuda100, 0., 1.);
    DDX_Text(pDX, IDC_BERMUDA200, m_bermuda200);
    DDV_MinMaxDouble(pDX, m_bermuda200, 0., 1.);
    DDX_Text(pDX, IDC_BERMUDA250, m_bermuda250);
    DDV_MinMaxDouble(pDX, m_bermuda250, 0., 1.);
    DDX_Text(pDX, IDC_BERMUDAR, m_bermudar);
    DDV_MinMaxDouble(pDX, m_bermudar, 0., 1.);
    DDX_Text(pDX, IDC_DALLIS, m_dallis);
    DDV_MinMaxDouble(pDX, m_dallis, 0., 1.);
    DDX_Text(pDX, IDC_FESCUE, m_fescue);
    DDV_MinMaxDouble(pDX, m_fescue, 0., 1.);
    DDX_Text(pDX, IDC_FESCUELAD, m_fescuelad);
    DDV_MinMaxDouble(pDX, m_fescuelad, 0., 1.);
    DDX_Text(pDX, IDC_KENTUCKY, m_kentucky);
    DDV_MinMaxDouble(pDX, m_kentucky, 0., 1.);
    DDX_Text(pDX, IDC_KENTUCKYW, m_kentuckyw);
    DDV_MinMaxDouble(pDX, m_kentuckyw, 0., 1.);
    DDX_Text(pDX, IDC_KUDZA, m_kudza);
    DDV_MinMaxDouble(pDX, m_kudza, 0., 1.);
    DDX_Text(pDX, IDC_LESPEDZA, m_lespedza);
    DDV_MinMaxDouble(pDX, m_lespedza, 0., 1.);
    DDX_Text(pDX, IDC_ORCHARD, m_orchard);
    DDV_MinMaxDouble(pDX, m_orchard, 0., 1.);
    DDX_Text(pDX, IDC_ORCHARDL, m_orchardl);
DDV_MinMaxDouble(pDX, m_orchardl, 0., 1.);
DDX_Text(pDX, IDC_ORCHARDR, m_orchardr);
DDV_MinMaxDouble(pDX, m_orchardr, 0., 1.);
DDX_Text(pDX, IDC_PEARL, m_pearl);
DDV_MinMaxDouble(pDX, m_pearl, 0., 1.);
DDX_Text(pDX, IDC_RYE, m_rye);
DDV_MinMaxDouble(pDX, m_rye, 0., 1.);
DDX_Text(pDX, IDC_SERICEA, m_sericea);
DDV_MinMaxDouble(pDX, m_sericea, 0., 1.);
DDX_Text(pDX, IDC_SUDAN, m_sudan);
DDV_MinMaxDouble(pDX, m_sudan, 0., 1.);
DDX_Text(pDX, IDC_SWITCH, m_switch);
DDV_MinMaxDouble(pDX, m_switch, 0., 1.);
DDX_Text(pDX, IDC_TOTAL, m_total);
DDV_MinMaxDouble(pDX, m_total, 0., 1.);
}
BEGIN_MESSAGE_MAP(CInput6, CDialog)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MENU, OnMenu)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

///////////////////////////////////////////////
// CInput6 message handlers
///////////////////////////////////////////////
CInput7::CInput7(CWnd* pParent /*=NULL*/)
: CDialog(CInput7::IDD, pParent)
{
    m_grcost1 = 0.0;
m_grcost10 = 0.0;
m_grcost11 = 0.0;
m_grcost12 = 0.0;
m_grcost13 = 0.0;
m_grcost14 = 0.0;
m_grcost15 = 0.0;
m_grcost16 = 0.0;
m_grcost17 = 0.0;
m_grcost18 = 0.0;
m_grcost19 = 0.0;
m_grcost2 = 0.0;
m_grcost20 = 0.0;
m_grcost3 = 0.0;
m_grcost4 = 0.0;
m_grcost5 = 0.0;
m_grcost6 = 0.0;
m_grcost7 = 0.0;
m_grcost8 = 0.0;
m_grcost9 = 0.0;
m_sopt1 = _T(" ");
m_sopt2 = _T(" ");
m_sopt3 = _T(""),
m_sopt4 = _T(""),
m_sopt5 = _T(""),
m_su1 = 0.0;
m_su10 = 0.0;
m_su11 = 0.0;
m_su12 = 0.0;
m_su13 = 0.0;
m_su14 = 0.0;
m_su15 = 0.0;
m_su16 = 0.0;
m_su17 = 0.0;
m_su18 = 0.0;
m_su19 = 0.0;
m_su2 = 0.0;
m_su20 = 0.0;
m_su3 = 0.0;
m_su4 = 0.0;
m_su5 = 0.0;
m_su6 = 0.0;
m_su7 = 0.0;
m_su8 = 0.0;
m_su9 = 0.0;
m_total = 0.0;
m_usesupp = FALSE;
}

void CInput7::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
DDX_Text(pDX, IDC_GRCOST1, m_grcost1);
DDV_MinMaxDouble(pDX, m_grcost1, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST10, m_grcost10);
DDV_MinMaxDouble(pDX, m_grcost10, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST11, m_grcost11);
DDV_MinMaxDouble(pDX, m_grcost11, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST12, m_grcost12);
DDV_MinMaxDouble(pDX, m_grcost12, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST13, m_grcost13);
DDV_MinMaxDouble(pDX, m_grcost13, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST14, m_grcost14);
DDV_MinMaxDouble(pDX, m_grcost14, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST15, m_grcost15);
DDV_MinMaxDouble(pDX, m_grcost15, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST16, m_grcost16);
DDV_MinMaxDouble(pDX, m_grcost16, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST17, m_grcost17);
DDV_MinMaxDouble(pDX, m_grcost17, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST18, m_grcost18);
DDV_MinMaxDouble(pDX, m_grcost18, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST19, m_grcost19);
DDV_MinMaxDouble(pDX, m_grcost19, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST2, m_grcost2);
DDV_MinMaxDouble(pDX, m_grcost2, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST3, m_grcost3);
DDV_MinMaxDouble(pDX, m_grcost3, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST4, m_grcost4);
DDV_MinMaxDouble(pDX, m_grcost4, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST5, m_grcost5);
DDV_MinMaxDouble(pDX, m_grcost5, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST6, m_grcost6);
DDV_MinMaxDouble(pDX, m_grcost6, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST7, m_grcost7);
DDV_MinMaxDouble(pDX, m_grcost7, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST8, m_grcost8);
DDV_MinMaxDouble(pDX, m_grcost8, 0., 9999.);
DDX_Text(pDX, IDC_GRCOST9, m_grcost9);
DDV_MinMaxDouble(pDX, m_grcost9, 0., 9999.);
DDX_Text(pDX, IDC_SOPT1, m_sopt1);
DDV_MaxChars(pDX, m_sopt1, 40);
DDX_Text(pDX, IDC_SOPT2, m_sopt2);
DDV_MaxChars(pDX, m_sopt2, 40);
DDX_Text(pDX, IDC_SOPT3, m_sopt3);
DDV_MaxChars(pDX, m_sopt3, 40);
DDX_Text(pDX, IDC_SOPT4, m_sopt4);
DDV_MaxChars(pDX, m_sopt4, 40);
DDX_Text(pDX, IDC_SOPT5, m_sopt5);
DDV_MaxChars(pDX, m_sopt5, 40);
DDX_Text(pDX, IDC_SU1, m_su1);
DDV_MinMaxDouble(pDX, m_su1, 0., 1.);
DDX_Text(pDX, IDC_SU10, m_su10);
DDV_MinMaxDouble(pDX, m_su10, 0., 1.);
DDX_Text(pDX, IDC_SU11, m_su11);
DDV_MinMaxDouble(pDX, m_su11, 0., 1.);
DDX_Text(pDX, IDC_SU12, m_su12);
DDV_MinMaxDouble(pDX, m_su12, 0., 1.);
DDX_Text(pDX, IDC_SU13, m_su13);
DDV_MinMaxDouble(pDX, m_su13, 0., 1.);
DDX_Text(pDX, IDC_SU14, m_su14);
DDV_MinMaxDouble(pDX, m_su14, 0., 1.);
DDX_Text(pDX, IDC_SU15, m_su15);
DDV_MinMaxDouble(pDX, m_su15, 0., 1.);
DDX_Text(pDX, IDC_SU16, m_su16);
DDV_MinMaxDouble(pDX, m_su16, 0., 1.);
DDX_Text(pDX, IDC_SU17, m_su17);
DDV_MinMaxDouble(pDX, m_su17, 0., 1.);
DDX_Text(pDX, IDC_SU18, m_su18);
DDV_MinMaxDouble(pDX, m_su18, 0., 1.);
DDX_Text(pDX, IDC_SU19, m_su19);
DDV_MinMaxDouble(pDX, m_su19, 0., 1.);
DDX_Text(pDX, IDC_SU2, m_su2);
DDV_MinMaxDouble(pDX, m_su2, 0., 1.);
DDX_Text(pDX, IDC_SU20, m_su20);
DDV_MinMaxDouble(pDX, m_su20, 0., 1.);
DDX_Text(pDX, IDC_SU3, m_su3);
DDV_MinMaxDouble(pDX, m_su3, 0., 1.);
DDX_Text(pDX, IDC_SU4, m_su4);
DDV_MinMaxDouble(pDX, m_su4, 0., 1.);
DDX_Text(pDX, IDC_SU5, m_su5);
DDV_MinMaxDouble(pDX, m_su5, 0., 1.);
DDX_Text(pDX, IDC_SU6, m_su6);
DDV_MinMaxDouble(pDX, m_su6, 0., 1.);
DDX_Text(pDX, IDC_SU7, m_su7);
DDV_MinMaxDouble(pDX, m_su7, 0., 1.);
DDX_Text(pDX, IDC_SU8, m_su8);
DDV_MinMaxDouble(pDX, m_su8, 0., 1.);
DDX_Text(pDX, IDC_SU9, m_su9);
DDV_MinMaxDouble(pDX, m_su9, 0., 1.);
DDX_Text(pDX, IDC_TOTAL, m_total);
DDV_MinMaxDouble(pDX, m_total, 0., 1.);
DDX_Check(pDX, IDC_USESUPP, m_usesupp);
}

BEGIN_MESSAGE_MAP(CInput7, CDialog)
ON_BN_CLICKED(IDC_MENU, OnMenu)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_BN_CLICKED(IDC_USESUPP, OnUsesupp)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////
// CInput7 message handlers
////////////////////////////////////////////////////////////////////////
CInput8::CInput8(CWnd* pParent /*=NULL*/)
    : CDialog(CInput8::IDD, pParent)
{
    m_sdm16 = 0.0;
    m_sdm17 = 0.0;
    m_sdm18 = 0.0;
    m_sdm19 = 0.0;
    m_sdm20 = 0.0;
    m_sme16 = 0.0;
    m_sme17 = 0.0;
    m_sme18 = 0.0;
    m_sme19 = 0.0;
    m_sme20 = 0.0;
    m_sneg16 = 0.0;
    m_sneg17 = 0.0;
    m_sneg18 = 0.0;
    m_sneg19 = 0.0;
    m_sneg20 = 0.0;
}
m_snem16 = 0.0;
m_snem17 = 0.0;
m_snem18 = 0.0;
m_snem19 = 0.0;
m_snem20 = 0.0;
m_sopt1 = _T(""");
m_sopt2 = _T(""");
m_sopt3 = _T(""");
m_sopt4 = _T(""");
m_sopt5 = _T(""");
m_stdn16 = 0.0;
m_stdn17 = 0.0;
m_stdn18 = 0.0;
m_stdn19 = 0.0;
m_stdn20 = 0.0;

}
DDV_MinMaxDouble(pDX, m_snem16, 0., 9999.);
DDX_Text(pDX, IDC_SNEM17, m_snem17);
DDV_MinMaxDouble(pDX, m_snem17, 0., 9999.);
DDX_Text(pDX, IDC_SNEM18, m_snem18);
DDV_MinMaxDouble(pDX, m_snem18, 0., 9999.);
DDX_Text(pDX, IDC_SNEM19, m_snem19);
DDV_MinMaxDouble(pDX, m_snem19, 0., 9999.);
DDX_Text(pDX, IDC_SNEM20, m_snem20);
DDV_MinMaxDouble(pDX, m_snem20, 0., 9999.);
DDX_Text(pDX, IDC_SOPT1, m_sopt1);
DDV_MaxChars(pDX, m_sopt1, 40);
DDX_Text(pDX, IDC_SOPT2, m_sopt2);
DDV_MaxChars(pDX, m_sopt2, 40);
DDX_Text(pDX, IDC_SOPT3, m_sopt3);
DDV_MaxChars(pDX, m_sopt3, 40);
DDX_Text(pDX, IDC_SOPT4, m_sopt4);
DDV_MaxChars(pDX, m_sopt4, 40);
DDX_Text(pDX, IDC_SOPT5, m_sopt5);
DDV_MaxChars(pDX, m_sopt5, 40);
DDX_Text(pDX, IDC_STDN16, m_stdn16);
DDV_MinMaxDouble(pDX, m_stdn16, 0., 9999.);
DDX_Text(pDX, IDC_STDN17, m_stdn17);
DDV_MinMaxDouble(pDX, m_stdn17, 0., 9999.);
DDX_Text(pDX, IDC_STDN18, m_stdn18);
DDVMinMaxDouble(pDX, m_stdn18, 0., 9999.);
DDX_Text(pDX, IDC_STDN19, m_stdn19);
DDV_MinMaxDouble(pDX, m_stdn19, 0., 9999.);
DDX_Text(pDX, IDC_STDN20, m_stdn20);
DDV_MinMaxDouble(pDX, m_stdn20, 0., 9999.);
}

BEGIN_MESSAGE_MAP(CInput8, CDialog)
ON_BN_CLICKED(IDC_MAIN, OnMain)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MAIN, OnPrev)
ON_BN_CLICKED(IDC_MAIN, OnPrev)
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CInput8, CDialog)
ON_BN_CLICKED(IDC_MAIN, OnMain)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MAIN, OnPrev)
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CInput8, CDialog)
ON_BN_CLICKED(IDC_MAIN, OnMain)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MAIN, OnPrev)
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CInput8, CDialog)
ON_BN_CLICKED(IDC_MAIN, OnMain)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MAIN, OnPrev)
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CInput8, CDialog)
ON_BN_CLICKED(IDC_MAIN, OnMain)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MAIN, OnPrev)
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CInput8, CDialog)
ON_BN_CLICKED(IDC_MAIN, OnMain)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MAIN, OnPrev)
END_MESSAGE_MAP()

BEGIN_MESSAGE_MAP(CInput8, CDialog)
ON_BN_CLICKED(IDC_MAIN, OnMain)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MAIN, OnPrev)
END_MESSAGE_MAP()
CDialog::EndDialog(IDC_MARKET);
}

void CMainMenu::OnNutrit()
{
CDialog::EndDialog(IDC_NUTRIT);
}

void CMainMenu::OnQuitsim()
{
CDialog::EndDialog(IDC_QUITSIM);
}

void CMainMenu::OnStartsim()
{
CDialog::EndDialog(IDC_STARTSIM);
}

void CInput2::OnImplantssteer()
{
CStatic* bmanytime = (CStatic *) GetDlgItem(IDC_SSSTATIC);
CEdit* emanytime = (CEdit *) GetDlgItem(IDC_SSEDIT);
if (IsDlgButtonChecked(IDC_IMPLANTSSTEER))
{
bmanytime->EnableWindow(TRUE);
emanytime->EnableWindow(TRUE);
}
else{
bmanytime->EnableWindow(FALSE);
emanytime->EnableWindow(FALSE);
}

void CInput2::OnImplantsheifer()
{
CStatic* bmanytime = (CStatic *) GetDlgItem(IDC_SHSTATIC);
CStatic* bday = (CStatic *) GetDlgItem(IDC_HBSDSTATIC);
CEdit* emonth = (CEdit *) GetDlgItem(IDC_EARLYMONTH);

CStatic* bmonth = (CStatic *) GetDlgItem(IDC_HBSSTATIC);
CStatic* bday = (CStatic *) GetDlgItem(IDC_HBSDSTATIC);
CEdit* emonth = (CEdit *) GetDlgItem(IDC_EARLYMONTH);
CEdit* eday = (CEdit *) GetDlgItem(IDC_EARLYDAY);
if (IsDlgButtonChecked(IDC_EARLYBS)) {
    bmonth->EnableWindow(TRUE);
    bday->EnableWindow(TRUE);
    emonth->EnableWindow(TRUE);
    eday->EnableWindow(TRUE);
}
else{
    bmonth->EnableWindow(FALSE);
    bday->EnableWindow(FALSE);
    emonth->EnableWindow(FALSE);
    eday->EnableWindow(FALSE);
}

void CInput2A::OnBred()
{

}

CStatic* pgroup = (CStatic *)GetDlgItem(IDC_PUREGROUP);
CStatic* pstatic = (CStatic *)GetDlgItem(IDC_PURESTATIC);
CComboBox* pedit = (CComboBox *)GetDlgItem(IDC_PURE);
CStatic* cgroup = (CStatic *)GetDlgItem(IDC_CROSSGROUP);
CStatic* cstatic = (CStatic *)GetDlgItem(IDC_CROSSSTATIC);
CStatic* cbreed = (CStatic *)GetDlgItem(IDC_BREEDSTATIC);
    CCheckListBox* ezcalf = (CCheckListBox
*)GetDlgItem(IDC_EZCALF);
    CCheckListBox* calfease = (CCheckListBox
*)GetDlgItem(IDC_CALFEASE);
    CComboBox* cedit = (CComboBox *)GetDlgItem(IDC_BULLBREED);
    CStatic* cezbbreed = (CStatic
*)GetDlgItem(IDC_EZBULLSTATIC);
    CComboBox* ezbull = (CComboBox
*)GetDlgItem(IDC_EZBULLBREED);
CStatic* s1 = (CStatic *)GetDlgItem(IDC_STATIC1);
CStatic* s2 = (CStatic *)GetDlgItem(IDC_STATIC2);
CStatic* s3 = (CStatic *)GetDlgItem(IDC_STATIC3);
CStatic* s4 = (CStatic *)GetDlgItem(IDC_STATIC4);
CStatic* s5 = (CStatic *)GetDlgItem(IDC_STATIC5);
CStatic* s6 = (CStatic *)GetDlgItem(IDC_STATIC6);
CStatic* s7 = (CStatic *)GetDlgItem(IDC_STATIC7);
CStatic* s8 = (CStatic *)GetDlgItem(IDC_STATIC8);
CStatic* s9 = (CStatic *)GetDlgItem(IDC_STATIC9);
CStatic* s10 = (CStatic *)GetDlgItem(IDC_STATIC10);
CStatic* s11 = (CStatic *)GetDlgItem(IDC_STATIC11);
CStatic* s12 = (CStatic *)GetDlgItem(IDC_STATIC12);
CEdit* c1 = (CEdit *)GetDlgItem(IDC_ANGUS);
CEdit* c2 = (CEdit *)GetDlgItem(IDC_HEREFORD);
CEdit* c3 = (CEdit *)GetDlgItem(IDC_PHEREFORD);
CEdit* c4 = (CEdit *)GetDlgItem(IDC_SHORTHORN);
CEdit* c5 = (CEdit *)GetDlgItem(IDC_CHAROL);
CEdit* c6 = (CEdit *)GetDlgItem(IDC_CHIANINA);
CEdit* c7 = (CEdit *)GetDlgItem(IDC_GELB);
CEdit* c8 = (CEdit *)GetDlgItem(IDC_LIMOUSIN);
CEdit* c9 = (CEdit *)GetDlgItem(IDC_MAINE);
CEdit* c10 = (CEdit *)GetDlgItem(IDC_PINZ);
CEdit* c11 = (CEdit *)GetDlgItem(IDC_SIMMENTAL);
CEdit* c12 = (CEdit *)GetDlgItem(IDC_TARENTAISE);
if (GetCheckedRadioButton(IDC_BRED, IDC_BRED)==IDC_BRED){
  pgroup->EnableWindow(TRUE);
  pstatic->EnableWindow(TRUE);
  pedit->EnableWindow(TRUE);
  cgroup->EnableWindow(FALSE);
  cstatic->EnableWindow(FALSE);
  cbreed->EnableWindow(FALSE);
  ezcalf->EnableWindow(FALSE);
  calfease->EnableWindow(FALSE);
  cedit->EnableWindow(FALSE);
  cezbbreed->EnableWindow(FALSE);
  ezbull->EnableWindow(FALSE);
  s1->EnableWindow(FALSE);
  s2->EnableWindow(FALSE);
  s3->EnableWindow(FALSE);
  s4->EnableWindow(FALSE);
  s5->EnableWindow(FALSE);
  s6->EnableWindow(FALSE);
  s7->EnableWindow(FALSE);
  s8->EnableWindow(FALSE);
  s9->EnableWindow(FALSE);
  s10->EnableWindow(FALSE);
  s11->EnableWindow(FALSE);
  s12->EnableWindow(FALSE);
  c1->EnableWindow(FALSE);
  c2->EnableWindow(FALSE);
  c3->EnableWindow(FALSE);
  c4->EnableWindow(FALSE);
  c5->EnableWindow(FALSE);
  c6->EnableWindow(FALSE);
  c7->EnableWindow(FALSE);
  c8->EnableWindow(FALSE);
  c9->EnableWindow(FALSE);
  c10->EnableWindow(FALSE);
  c11->EnableWindow(FALSE);
  c12->EnableWindow(FALSE);
}
if (GetCheckedRadioButton(IDC_CROSSBRED, IDC_CROSSBRED)==IDC_CROSSBRED){
  pgroup->EnableWindow(FALSE);
  pstatic->EnableWindow(FALSE);
  pedit->EnableWindow(FALSE);
  cgroup->EnableWindow(TRUE);
  cstatic->EnableWindow(TRUE);
  cbreed->EnableWindow(TRUE);
  ezcalf->EnableWindow(TRUE);
void CInput2A::OnEzCalf()
{
    CStatic* eztext = (CStatic *) GetDlgItem(IDC_EZBULLSTATIC);
    CComboBox* ezbox = (CComboBox *) GetDlgItem(IDC_EZBULLBREED);
    if (IsDlgButtonChecked(IDC_EZCALF)){
        eztext->EnableWindow(TRUE);
        ezbox->EnableWindow(TRUE);
    } else{
        eztext->EnableWindow(FALSE);
        ezbox->EnableWindow(FALSE);
    }
}

void CInput1::OnNext()
{
    UpdateData();
    CDialog::EndDialog(IDC_NEXT);
}
void CInput1::OnMenu()
{
    UpdateData();
    CDialog::EndDialog(IDC_MENU);
}

void CInput2::OnNext()
{
    UpdateData();
    CDialog::EndDialog(IDC_NEXT);
}

void CInput2::OnMenu()
{
    UpdateData();
    CDialog::EndDialog(IDC_MENU);
}

void CInput2::OnPrev()
{
    UpdateData();
    CDialog::EndDialog(IDC_PREV);
}

void CInput2::Startup()
{
    OnImplantsheifer();
    OnImplantssteer();
    OnEarlybs();
}

BOOL CInput2::OnInitDialog()
{
    CDialog::OnInitDialog();
    if (startrefresh == TRUE){
        OnBuycalves();
        OnImplantsheifer();
        OnImplantssteer();
        startrefresh=FALSE;
    }
    return TRUE;
}

void CInput2::OnBuycalves()
{
    CStatic* reptext = (CStatic *) GetDlgItem(IDC_REPSTATIC);
    CComboBox* repbox = (CComboBox *) GetDlgItem(IDC_REPBREED);
    CButton* radio1 = (CButton *) GetDlgItem(IDC_REPPURE);
    CButton* radio2 = (CButton *) GetDlgItem(IDC_RADIO2);
    if  (IsDlgButtonChecked(IDC_BUYCALVES)){
        reptext->EnableWindow(TRUE);
        repbox->EnableWindow(TRUE);
    }
radio1->EnableWindow(TRUE);
radio2->EnableWindow(TRUE);
} else if (IsDlgButtonChecked(IDC_BUYPREG)){
    reptext->EnableWindow(TRUE);
    repbox->EnableWindow(TRUE);
    radio1->EnableWindow(TRUE);
    radio2->EnableWindow(TRUE);
} else{
    reptext->EnableWindow(FALSE);
    repbox->EnableWindow(FALSE);
    radio1->EnableWindow(FALSE);
    radio2->EnableWindow(FALSE);
}
}

void CInput2A::OnNext()
{
    UpdateData();
    CDialog::EndDialog(IDC_NEXT);
}

BOOL CInput2A::OnInitDialog()
{
    CDialog::OnInitDialog();
    OnBred();
    OnEzCalf();
    return TRUE;
}

void CInput2A::OnPrev()
{
    UpdateData();
    CDialog::EndDialog(IDC_PREV);
}

void CInput2A::OnMenu()
{
    UpdateData();
    CDialog::EndDialog(IDC_MENU);
}

void CInput2B::CInput2B(CWnd* pParent /*=NULL*/)
    : CDialog(CInput2B::IDD, pParent)
{
    m_fallday = 0;
    m_falllength = 0;
    m_fallmonth = 0;
    m_shbse = FALSE;
}
m_shbse2 = FALSE;
m_shbse2day = 0;
m_shbse2month = 0;
m_shbseday = 0;
m_shbsemonth = 0;
m_breedingseasons = -1;
m_springday = 0;
m_springlength = 0;
m_springmonth = 0;
m_weansday = 0;
m_weansmonth = 0;
m_weanfday = 0;
m_weanfmonth = 0;
}
void CInput2B::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
DDX_Text(pDX, IDC_FALLDAY, m_fallday);
DDV_MinMaxInt(pDX, m_fallday, 1, 31);
DDX_Text(pDX, IDC_FALLLENGTH, m_falllength);
DDV_MinMaxInt(pDX, m_falllength, 1, 120);
DDX_Text(pDX, IDC_FALLMONTH, m_fallmonth);
DDV_MinMaxInt(pDX, m_fallmonth, 1, 12);
DDX_Text(pDX, IDC_SHBSE2DAY, m_shbse2day);
DDX_MinMaxInt(pDX, pDX, m_shbse2day, 1, 31);
DDX_Text(pDX, IDC_SHBSE2MONTH, m_shbse2month);
DDX_MinMaxInt(pDX, pDX, m_shbse2month, 1, 12);
DDX_Text(pDX, IDC_SHBSEDAY, m_shbseday);
DDX_MinMaxInt(pDX, pDX, m_shbseday, 1, 31);
DDX_Text(pDX, IDC_SHBSEMONTH, m_shbsemonth);
DDX_MinMaxInt(pDX, pDX, m_shbsemonth, 1, 12);
DDX_Radio(pDX, IDC_SPRINGCALVING, m_breedingseasons);
DDX_Text(pDX, IDC_SPRINGDAY, m_springday);
DDV_MinMaxInt(pDX, m_springday, 1, 31);
DDX_Text(pDX, IDC_SPRINGLENGTH, m_springlength);
DDV_MinMaxInt(pDX, m_springlength, 1, 120);
DDX_Text(pDX, IDC_SPRINGMONTH, m_springmonth);
DDV_MinMaxInt(pDX, m_springmonth, 1, 12);
DDX_Text(pDX, IDC_WEANSDAY, m_weansday);
DDX_MinMaxInt(pDX, pDX, m_weansday, 1, 31);
DDX_Text(pDX, IDC_WEANSMONTH, m_weansmonth);
DDX_MinMaxInt(pDX, pDX, m_weansmonth, 1, 12);
DDX_Text(pDX, IDC_WEANFDAY, m_weanfday);
DDV_MinMaxInt(pDX, m_weanfday, 1, 31);
DDX_Text(pDX, IDC_WEANFMONTH, m_weanfmonth);
DDV_MinMaxInt(pDX, pDX, m_weanfmonth, 1, 12);
}
BEGIN_MESSAGE_MAP(CInput2B, CDialog)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_MENU, OnMenu)
}
ON_BN_CLICKED(IDC_SPRINGCALVING, OnSpringcalving)
ON_BN_CLICKED(IDC_FALLCALVING, OnSpringcalving)
ON_BN_CLICKED(IDC_BOTHCALVING, OnSpringcalving)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

void CInput2B::OnPrev()
{
UpdateData();
CDialog::EndDialog(IDC_PREV);
}

void CInput2B::OnMenu()
{
UpdateData();
CDialog::EndDialog(IDC_MENU);
}

void CInput2B::OnSpringcalving()
{
CStatic* springbox = (CStatic *) GetDlgItem(IDC_SPRINGBOX);
CStatic* springstatic = (CStatic *) GetDlgItem(IDC_SPRINGSTATIC);
CStatic* springstatic2 = (CStatic *) GetDlgItem(IDC_SPRINGSTATIC2);
CStatic* springsmonth = (CStatic *) GetDlgItem(IDC_SPRINGSMONTH);
CStatic* springsday = (CStatic *) GetDlgItem(IDC_SPRINGSDAY);
CEdit* springmonth = (CEdit *) GetDlgItem(IDC_SPRINGMONTH);
CEdit* springday = (CEdit *) GetDlgItem(IDC_SPRINGDAY);
CEdit* springlength = (CEdit *) GetDlgItem(IDC_SPRINGLENGTH);
CStatic* weanmonth = (CStatic *) GetDlgItem(IDC_SWEANMONTH);
CStatic* weanday = (CStatic *) GetDlgItem(IDC_SWEANDAY);
CEdit* weansmonth = (CEdit *) GetDlgItem(IDC_WEANSMONTH);
CEdit* weansday = (CEdit *) GetDlgItem(IDC_WEANSDAY);
CStatic* shbsestatic = (CStatic *) GetDlgItem(IDC_SHBSESTATIC);
CStatic* shbsestatic2 = (CStatic *) GetDlgItem(IDC_SHBSESTATIC2);
CStatic* shbse2static = (CStatic *) GetDlgItem(IDC_SHBSE2STATIC);
CStatic* shbse2static2 = (CStatic *) GetDlgItem(IDC_SHBSE2STATIC2);
CCheckListBox* shbse = (CCheckListBox *) GetDlgItem(IDC_SHBSE);
CEdit* shbsemonth = (CEdit *) GetDlgItem(IDC_SHBSEMONTH);
CEdit* shbseday = (CEdit *) GetDlgItem(IDC_SHBSEDAY);
CStatic* fallbox = (CStatic *) GetDlgItem(IDC_FALLBOX);
CStatic* fallstatic = (CStatic *) GetDlgItem(IDC_FALLSTATIC);
CStatic* fallstatic2 = (CStatic *) GetDlgItem(IDC_FALLSTATIC2);
CStatic* fallsmonth = (CStatic *) GetDlgItem(IDC_FALLSMONTH);
CStatic* fallsday = (CStatic *) GetDlgItem(IDC_FALLSDAY);
CEdit* fallmonth = (CEdit *) GetDlgItem(IDC_FALLMONTH);
CEdit* fallday = (CEdit *) GetDlgItem(IDC_FALLDAY);
CEdit* falllength = (CEdit *) GetDlgItem(IDC_FALLLENGTH);
CStatic* fweanmonth = (CStatic *) GetDlgItem(IDC_FWEANMONTH);
CStatic* fweanday = (CStatic *) GetDlgItem(IDC_FWEANDAY);
CEdit* weanmonth = (CEdit *) GetDlgItem(IDC_WEANMONTH);
CEdit* weanday = (CEdit *) GetDlgItem(IDC_WEANDAY);
CEdit* weanlength = (CEdit *) GetDlgItem(IDC_WEANLENGTH);
CStatic* fallheifstatic = (CStatic *) GetDlgItem(IDC_FALLHEIFSTATIC);
CStatic* springheifstatic = (CStatic *) GetDlgItem(IDC_SPRINGHEIFSTATIC);
CEdit* shbse2month = (CEdit *) GetDlgItem(IDC_SHBSE2MONTH);
CEdit* shbse2day = (CEdit *) GetDlgItem(IDC_SHBSE2DAY);
CStatic* fallewesstatic = (CStatic *) GetDlgItem(IDC_FALLWEANSTATIC);
CStatic* springsstatic = (CStatic *) GetDlgItem(IDC_SPRINGWEANSTATIC);
if (GetCheckedRadioButton(IDC_SPRINGCALVING, IDC_SPRINGCALVING) == IDC_SPRINGCALVING) {
    springbox->EnableWindow(TRUE);
    springbox2->EnableWindow(TRUE);
    springsmonth->EnableWindow(TRUE);
    springsday->EnableWindow(TRUE);
    springmonth->EnableWindow(TRUE);
    springday->EnableWindow(TRUE);
    springlength->EnableWindow(TRUE);
    springstatic->EnableWindow(TRUE);
    springstatic2->EnableWindow(TRUE);
    shbsestatic->EnableWindow(TRUE);
    shbsestatic2->EnableWindow(TRUE);
    weanmonth->EnableWindow(TRUE);
    weanday->EnableWindow(TRUE);
    weansday->EnableWindow(TRUE);
    weanmonth->EnableWindow(TRUE);
    springwesstatic->EnableWindow(TRUE);
    OnShbse();
    fallbox->EnableWindow(FALSE);
    fallbox2->EnableWindow(FALSE);
    fallsmonth->EnableWindow(FALSE);
fallsday->EnableWindow(FALSE);
fallmonth->EnableWindow(FALSE);
fallday->EnableWindow(FALSE);
falllength->EnableWindow(FALSE);
fallstatic->EnableWindow(FALSE);
fallheifstatic->EnableWindow(FALSE);
shbse2month->EnableWindow(FALSE);
shbse2day->EnableWindow(FALSE);
shbse2static->EnableWindow(FALSE);
shbse2static2->EnableWindow(FALSE);
fweanmonth->EnableWindow(FALSE);
fweanday->EnableWindow(FALSE);
weanfday->EnableWindow(FALSE);
weanfmonth->EnableWindow(FALSE);
fallweanstatic->EnableWindow(FALSE);
}
else if (GetCheckedRadioButton(IDC_FALLCALVING,
IDC_FALLCALVING)==IDC_FALLCALVING)
{
fallbox->EnableWindow(TRUE);
fallbox->EnableWindow(TRUE);
fallmonth->EnableWindow(TRUE);
fallsday->EnableWindow(TRUE);
fallday->EnableWindow(TRUE);
fallmonth->EnableWindow(TRUE);
falllength->EnableWindow(TRUE);
fallstatic->EnableWindow(TRUE);
fallstatic2->EnableWindow(TRUE);
fweanmonth->EnableWindow(TRUE);
fweanday->EnableWindow(TRUE);
weanfday->EnableWindow(TRUE);
weanfmonth->EnableWindow(TRUE);
fallweanstatic->EnableWindow(TRUE);
shbsestatic->EnableWindow(FALSE);
shbsestatic2->EnableWindow(FALSE);
shbse2static->EnableWindow(TRUE);
shbse2static2->EnableWindow(TRUE);
OnShbse2();
springbox->EnableWindow(FALSE);
springbox->EnableWindow(FALSE);
springsmonth->EnableWindow(FALSE);
springsday->EnableWindow(FALSE);
springmonth->EnableWindow(FALSE);
springday->EnableWindow(FALSE);
springlength->EnableWindow(FALSE);
springstatic->EnableWindow(FALSE);
springstatic2->EnableWindow(FALSE);
springheifstatic->EnableWindow(FALSE);
shbsemonth->EnableWindow(FALSE);
shbseday->EnableWindow(FALSE);
BOOL CInput2B::OnInitDialog()
{
    CDialo::OnInitDialog()
    OnShbse();
    OnShbse2();
    OnSpringcalving();
}

springweanstatic->EnableWindow(FALSE);
weanmonth->EnableWindow(FALSE);
weanday->EnableWindow(FALSE);
weansday->EnableWindow(FALSE);
weansmonth->EnableWindow(FALSE);
}
else if (GetCheckedRadioButton(IDC_BOTHCALVING, IDC_BOTHCALVING)==IDC_BOTHCALVING)
{
springbox->EnableWindow(TRUE);
springbox->EnableWindow(TRUE);
springsmonth->EnableWindow(TRUE);
springsday->EnableWindow(TRUE);
springday->EnableWindow(TRUE);
springmonth->EnableWindow(TRUE);
springlength->EnableWindow(TRUE);
springstatic->EnableWindow(TRUE);
springstatic2->EnableWindow(TRUE);
weanmonth->EnableWindow(TRUE);
weanday->EnableWindow(TRUE);
weansday->EnableWindow(TRUE);
weansmonth->EnableWindow(TRUE);
springweanstatic->EnableWindow(TRUE);
shbsestatic->EnableWindow(TRUE);
shbsestatic2->EnableWindow(TRUE);
shbse2static->EnableWindow(TRUE);
shbse2static2->EnableWindow(TRUE);
fallback->EnableWindow(TRUE);
fallback->EnableWindow(TRUE);
fallsmmoth->EnableWindow(TRUE);
fallsday->EnableWindow(TRUE);
fallday->EnableWindow(TRUE);
fallmonth->EnableWindow(TRUE);
falllength->EnableWindow(TRUE);
fallstatic->EnableWindow(TRUE);
fallstatic2->EnableWindow(TRUE);
fweanmonth->EnableWindow(TRUE);
fweanday->EnableWindow(TRUE);
weanfday->EnableWindow(TRUE);
weanfmonth->EnableWindow(TRUE);
fallweanstatic->EnableWindow(TRUE);
OnShbse();
OnShbse2();
}
return TRUE;
}
void CInput2B::OnShbse2()
{
CEdit* fallheifstatic = (CEdit *) GetDlgItem(IDC_FALLHEIFSTATIC);
CEdit* shbse2month = (CEdit *) GetDlgItem(IDC_SHBSE2MONTH);
CEdit* shbse2day = (CEdit *) GetDlgItem(IDC_SHBSE2DAY);
fallheifstatic->EnableWindow(TRUE);
shbse2month->EnableWindow(TRUE);
shbse2day->EnableWindow(TRUE);
}
void CInput2B::OnShbse()
{
CEdit* springheifstatic = (CEdit *) GetDlgItem(IDC_SPRINGHEIFSTATIC);
CEdit* shbsemonth = (CEdit *) GetDlgItem(IDC_SHBSEMONTH);
CEdit* shbseday = (CEdit *) GetDlgItem(IDC_SHBSEDAY);
springheifstatic->EnableWindow(TRUE);
shbsemonth->EnableWindow(TRUE);
shbseday->EnableWindow(TRUE);
}
void CInput3::OnNext()
{
UpdateData();
CDialog::EndDialog(IDC_NEXT);
}
void CInput3::OnMenu()
{
UpdateData();
CDialog::EndDialog(IDC_MENU);
}
void CInput3::Startup()
{
OnHedgecalves();
OnMarketingcattle();
}
BOOL CInput3::OnInitDialog()
{
CDialog::OnInitDialog();
if (startrefresh == TRUE){
OnHedgecalves();
OnMarketingcattle();
startrefresh=FALSE;
}
return TRUE;
void CInput3::OnHedgecalves()
{
    CComboBox* hedg1 = (CComboBox *) GetDlgItem(IDC_HEDG1);
    CComboBox* hedg2 = (CComboBox *) GetDlgItem(IDC_HEDG2);
    CStatic* willhedge = (CStatic *) GetDlgItem(IDC_STATICH);
    CEdit* difference = (CEdit *) GetDlgItem(IDC_DIFF);
    if (IsDlgButtonChecked(IDC_HEDG1))
    {
        willhedge->EnableWindow(TRUE);
        difference->EnableWindow(TRUE);
    } else if (IsDlgButtonChecked(IDC_HEDG2))
    {
        willhedge->EnableWindow(TRUE);
        difference->EnableWindow(TRUE);
    }
    else
    {
        willhedge->EnableWindow(FALSE);
        difference->EnableWindow(FALSE);
    }
}

void CInput3::OnMarketingcattle()
{
    CButton* radio2 = (CButton *) GetDlgItem(IDC_RADIO2);
    CButton* radio3 = (CButton *) GetDlgItem(IDC_RADIO3);
    CButton* radio4 = (CButton *) GetDlgItem(IDC_RADIO4);
    CButton* radio5 = (CButton *) GetDlgItem(IDC_RADIO5);
    CButton* radio7 = (CButton *) GetDlgItem(IDC_RADIO7);
    CButton* radio8 = (CButton *) GetDlgItem(IDC_RADIO8);
    CButton* radio10 = (CButton *) GetDlgItem(IDC_RADIO10);
    CButton* radio11 = (CButton *) GetDlgItem(IDC_RADIO11);
    CButton* radio13 = (CButton *) GetDlgItem(IDC_RADIO13);
    CButton* radio14 = (CButton *) GetDlgItem(IDC_RADIO14);
    CButton* radioorphan = (CButton *) GetDlgItem(IDC_ORPHAN);
    CButton* radioculled = (CButton *) GetDlgItem(IDC_CULLED);
    CButton* radioheifer = (CButton *) GetDlgItem(IDC_HEIFER);
    CButton* radiossteer = (CButton *) GetDlgItem(IDC_SSTEER);
    CButton* radiosheifer = (CButton *) GetDlgItem(IDC_SHEIFER);
    CStatic* willtravel = (CStatic *) GetDlgItem(IDC_STATICD);
    CEdit* distance = (CEdit *) GetDlgItem(IDC_DIST);
    CComboBox* overnight = (CComboBox *) GetDlgItem(IDC_ONITE);
    if ( (GetCheckedRadioButton(IDC_RADIO2,IDC_RADIO2)!=IDC_RADIO2) &&
          (GetCheckedRadioButton(IDC_RADIO3,IDC_RADIO3)!=IDC_RADIO3) &&
          (GetCheckedRadioButton(IDC_RADIO4,IDC_RADIO4)!=IDC_RADIO4) &&
          (GetCheckedRadioButton(IDC_RADIO5,IDC_RADIO5)!=IDC_RADIO5) &&
(GetCheckedRadioButton(IDC_RADIO7, IDC_RADIO7) != IDC_RADIO7)
&&
(GetCheckedRadioButton(IDC_RADIO8, IDC_RADIO8) != IDC_RADIO8)
&&
(GetCheckedRadioButton(IDC_RADIO10, IDC_RADIO10) != IDC_RADIO10) &&
(GetCheckedRadioButton(IDC_RADIO11, IDC_RADIO11) != IDC_RADIO11) &&
(GetCheckedRadioButton(IDC_RADIO13, IDC_RADIO13) != IDC_RADIO13) &&
(GetCheckedRadioButton(IDC_RADIO14, IDC_RADIO14) != IDC_RADIO14) ) {
    willtravel->EnableWindow(FALSE);
distance->EnableWindow(FALSE);
overnight->EnableWindow(FALSE);
}
else if ( (GetCheckedRadioButton(IDC_RADIO2, IDC_RADIO2) ==
IDC_RADIO2) ||
(GetCheckedRadioButton(IDC_RADIO3, IDC_RADIO3) ==
IDC_RADIO3) ||
(GetCheckedRadioButton(IDC_RADIO4, IDC_RADIO4) ==
IDC_RADIO4) ||
(GetCheckedRadioButton(IDC_RADIO5, IDC_RADIO5) ==
IDC_RADIO5) ||
(GetCheckedRadioButton(IDC_RADIO7, IDC_RADIO7) ==
IDC_RADIO7) ||
(GetCheckedRadioButton(IDC_RADIO8, IDC_RADIO8) ==
IDC_RADIO8) ||
(GetCheckedRadioButton(IDC_RADIO10, IDC_RADIO10) ==
IDC_RADIO10) ||
(GetCheckedRadioButton(IDC_RADIO11, IDC_RADIO11) ==
IDC_RADIO11) ||
(GetCheckedRadioButton(IDC_RADIO13, IDC_RADIO13) ==
IDC_RADIO13) ||
(GetCheckedRadioButton(IDC_RADIO14, IDC_RADIO14) ==
IDC_RADIO14) ) {
    willtravel->EnableWindow(TRUE);
distance->EnableWindow(TRUE);
overnight->EnableWindow(TRUE);
}
else if ( (GetCheckedRadioButton(IDC_ORPHAN, IDC_ORPHAN) == IDC_ORPHAN) &&
(GetCheckedRadioButton(IDC_CULLED, IDC_CULLED) == IDC_CULLED) &&
(GetCheckedRadioButton(IDC_STEER, IDC_STEER) == IDC_STEER) &&
(GetCheckedRadioButton(IDC_HEIFER, IDC_HEIFER) == IDC_HEIFER) &&
(GetCheckedRadioButton(IDC_SSTEER, IDC_SSTEER) == IDC_SSTEER) &&
(GetCheckedRadioButton(IDC_SHEIFER, IDC_SHEIFER) == IDC_SHEIFER) ) {


willtravel->EnableWindow(FALSE);
distance->EnableWindow(FALSE);
overnight->EnableWindow(FALSE);
}
}

void CInput4::OnPrev()
{
UpdateData();
CDialog::EndDialog(IDC_PREV);
}

void CInput4::OnMenu()
{
UpdateData();
CDialog::EndDialog(IDC_MENU);
}

void CInput5::OnNext()
{
UpdateData();
CDialog::EndDialog(IDC_NEXT);
}

void CInput5::OnMenu()
{
UpdateData();
CDialog::EndDialog(IDC_MENU);
}

void CInput6::OnPrev()
{
UpdateData();
CDialog::EndDialog(IDC_PREV);
}

void CInput6::OnMenu()
{
UpdateData();
CDialog::EndDialog(IDC_MENU);
}

void CInput7::OnMenu()
{
UpdateData();
CDialog::EndDialog(IDC_MENU);
}

void CInput7::OnNext()
{
UpdateData();
}
void CInput8::OnMain()
{
  UpdateData();
  CDialog::EndDialog(IDC_MENU);
}

void CInput8::OnPrev()
{
  UpdateData();
  CDialog::EndDialog(IDC_PREV);
}

/////////////////////////////////////////////////
YearOption::YearOption(CWnd* pParent /*=NULL*/)
: CDialog(YearOption::IDD, pParent)
{
  m_changesire = FALSE;
  m_sire = 0;
  m_changeezsire = FALSE;
  m_sire2 = 0;
  m_caption = _T("");  
}

void YearOption::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  DDX_Check(pDX, IDC_CHANGESIRE, m_changesire);
  DDX_CBIndex(pDX, IDC_SIRE, m_sire);
  DDX_Check(pDX, IDC_CHANGEEZSIRE, m_changeezsire);
  DDX_CBIndex(pDX, IDC_SIRE2, m_sire2);
  DDX_Text(pDX, IDC_YEARX, m_caption);
  DDV_MaxChars(pDX, m_caption, 40);
}

BEGIN_MESSAGE_MAP(YearOption, CDialog)
ON_BN_CLICKED(IDC_CHANGESIRE, OnChangesire)
ON_BN_CLICKED(IDC_CHANGEEZSIRE, OnChangesire)
ON_BN_CLICKED(ID_HELP_CONTENTS, OnHelpContents)
END_MESSAGE_MAP()

///////////////////////////////////////////////////////////////////////
void YearOption::OnChangesire()
{
  CListBox* sire = (CListBox *) GetDlgItem(IDC_SIRE);
  CListBox* ezsire = (CListBox *) GetDlgItem(IDC_SIRE2);
  if (IsDlgButtonChecked(IDC_CHANGESIRE))
  {
    sire->EnableWindow(TRUE);
    ezsire->EnableWindow(FALSE);
  }
  else
  {
    sire->EnableWindow(FALSE);
    ezsire->EnableWindow(TRUE);
  }
}
} 
else{
    sire->EnableWindow(FALSE);
}
if (IsDlgButtonChecked(IDC_CHANGEEZRRE)) {
    ezsire->EnableWindow(TRUE);
} 
else{
    ezsire->EnableWindow(FALSE);
}
}

BOOL YearOption::OnInitDialog()
{
    CDialog::OnInitDialog();
    OnChangesire();
    return TRUE;
}

void CInput7::OnUsesupp()
{
    CEdit* gr1 = (CEdit *) GetDlgItem(IDC_GRCOST1);
    CEdit* gr2 = (CEdit *) GetDlgItem(IDC_GRCOST2);
    CEdit* gr3 = (CEdit *) GetDlgItem(IDC_GRCOST3);
    CEdit* gr4 = (CEdit *) GetDlgItem(IDC_GRCOST4);
    CEdit* gr5 = (CEdit *) GetDlgItem(IDC_GRCOST5);
    CEdit* gr6 = (CEdit *) GetDlgItem(IDC_GRCOST6);
    CEdit* gr7 = (CEdit *) GetDlgItem(IDC_GRCOST7);
    CEdit* gr8 = (CEdit *) GetDlgItem(IDC_GRCOST8);
    CEdit* gr9 = (CEdit *) GetDlgItem(IDC_GRCOST9);
    CEdit* gr10 = (CEdit *) GetDlgItem(IDC_GRCOST10);
    CEdit* gr11 = (CEdit *) GetDlgItem(IDC_GRCOST11);
    CEdit* gr12 = (CEdit *) GetDlgItem(IDC_GRCOST12);
    CEdit* gr13 = (CEdit *) GetDlgItem(IDC_GRCOST13);
    CEdit* gr14 = (CEdit *) GetDlgItem(IDC_GRCOST14);
    CEdit* gr15 = (CEdit *) GetDlgItem(IDC_GRCOST15);
    CEdit* gr16 = (CEdit *) GetDlgItem(IDC_GRCOST16);
    CEdit* gr17 = (CEdit *) GetDlgItem(IDC_GRCOST17);
    CEdit* gr18 = (CEdit *) GetDlgItem(IDC_GRCOST18);
    CEdit* gr19 = (CEdit *) GetDlgItem(IDC_GRCOST19);
    CEdit* gr20 = (CEdit *) GetDlgItem(IDC_GRCOST20);
    CButton* next = (CButton *) GetDlgItem(IDC_NEXT);
    CEdit* so1 = (CEdit *) GetDlgItem(IDC_SOPT1);
    CEdit* so2 = (CEdit *) GetDlgItem(IDC_SOPT2);
    CEdit* so3 = (CEdit *) GetDlgItem(IDC_SOPT3);
    CEdit* so4 = (CEdit *) GetDlgItem(IDC_SOPT4);
    CEdit* so5 = (CEdit *) GetDlgItem(IDC_SOPT5);
    CEdit* su1 = (CEdit *) GetDlgItem(IDC_SU1);
    CEdit* su2 = (CEdit *) GetDlgItem(IDC_SU2);
    CEdit* su3 = (CEdit *) GetDlgItem(IDC_SU3);
    CEdit* su4 = (CEdit *) GetDlgItem(IDC_SU4);
CEdit* su5 = (CEdit *) GetDlgItem(IDC_SU5);
CEdit* su6 = (CEdit *) GetDlgItem(IDC_SU6);
CEdit* su7 = (CEdit *) GetDlgItem(IDC_SU7);
CEdit* su8 = (CEdit *) GetDlgItem(IDC_SU8);
CEdit* su9 = (CEdit *) GetDlgItem(IDC_SU9);
CEdit* su10 = (CEdit *) GetDlgItem(IDC_SU10);
CEdit* su11 = (CEdit *) GetDlgItem(IDC_SU11);
CEdit* su12 = (CEdit *) GetDlgItem(IDC_SU12);
CEdit* su13 = (CEdit *) GetDlgItem(IDC_SU13);
CEdit* su14 = (CEdit *) GetDlgItem(IDC_SU14);
CEdit* su15 = (CEdit *) GetDlgItem(IDC_SU15);
CEdit* su16 = (CEdit *) GetDlgItem(IDC_SU16);
CEdit* su17 = (CEdit *) GetDlgItem(IDC_SU17);
CEdit* su18 = (CEdit *) GetDlgItem(IDC_SU18);
CEdit* su19 = (CEdit *) GetDlgItem(IDC_SU19);
CEdit* su20 = (CEdit *) GetDlgItem(IDC_SU20);
CEdit* total = (CEdit *) GetDlgItem(IDC_TOTAL);
if (IsDlgButtonChecked(IDC_USESUPP))
{
gr1->EnableWindow(true);
gr2->EnableWindow(true);
gr3->EnableWindow(true);
gr4->EnableWindow(true);
gr5->EnableWindow(true);
gr6->EnableWindow(true);
gr7->EnableWindow(true);
gr8->EnableWindow(true);
gr9->EnableWindow(true);
gr10->EnableWindow(true);
gr11->EnableWindow(true);
gr12->EnableWindow(true);
gr13->EnableWindow(true);
gr14->EnableWindow(true);
gr15->EnableWindow(true);
gr16->EnableWindow(true);
gr17->EnableWindow(true);
gr18->EnableWindow(true);
gr19->EnableWindow(true);
gr20->EnableWindow(true);
next->EnableWindow(true);
sol->EnableWindow(true);
sol2->EnableWindow(true);
sol3->EnableWindow(true);
sol4->EnableWindow(true);
sol5->EnableWindow(true);
sul1->EnableWindow(true);
sul2->EnableWindow(true);
sul3->EnableWindow(true);
sul4->EnableWindow(true);
sul5->EnableWindow(true);
sul6->EnableWindow(true);
sul7->EnableWindow(true);
su8->EnableWindow(true);
su9->EnableWindow(true);
su10->EnableWindow(true);
su11->EnableWindow(true);
su12->EnableWindow(true);
su13->EnableWindow(true);
su14->EnableWindow(true);
su15->EnableWindow(true);
su16->EnableWindow(true);
su17->EnableWindow(true);
su18->EnableWindow(true);
su19->EnableWindow(true);
su20->EnableWindow(true);
total->EnableWindow(true);
}
else{
gr1->EnableWindow(false);
gr2->EnableWindow(false);
gr3->EnableWindow(false);
gr4->EnableWindow(false);
gr5->EnableWindow(false);
gr6->EnableWindow(false);
gr7->EnableWindow(false);
gr8->EnableWindow(false);
gr9->EnableWindow(false);
gr10->EnableWindow(false);
gr11->EnableWindow(false);
gr12->EnableWindow(false);
gr13->EnableWindow(false);
gr14->EnableWindow(false);
gr15->EnableWindow(false);
gr16->EnableWindow(false);
gr17->EnableWindow(false);
gr18->EnableWindow(false);
gr19->EnableWindow(false);
gr20->EnableWindow(false);
next->EnableWindow(false);
sol->EnableWindow(false);
s02->EnableWindow(false);
s03->EnableWindow(false);
s04->EnableWindow(false);
s05->EnableWindow(false);
su1->EnableWindow(false);
su2->EnableWindow(false);
su3->EnableWindow(false);
su4->EnableWindow(false);
su5->EnableWindow(false);
su6->EnableWindow(false);
su7->EnableWindow(false);
su8->EnableWindow(false);
su9->EnableWindow(false);
su10->EnableWindow(false);
su11->EnableWindow(false);
su12->EnableWindow(false);
su13->EnableWindow(false);
su14->EnableWindow(false);
su15->EnableWindow(false);
su16->EnableWindow(false);
su17->EnableWindow(false);
su18->EnableWindow(false);
su19->EnableWindow(false);
su20->EnableWindow(false);
total->EnableWindow(false);
}

BOOL CInput7::OnInitDialog()
{
    CDialog::OnInitDialog();
    OnUsesupp();
    return TRUE;
}

BOOL CDieHerd::OnInitDialog()
{
    CDialog::OnInitDialog();
    return TRUE;
}

void CDieHerd::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_PROB, m_prob);
    DDV_MaxChars(pDX, m_prob, 40);
}

BEGIN_MESSAGE_MAP(CDieHerd, CDialog)
    // DieHerd dialog
    COMMAND_ID, _T(""), m_prob
END_MESSAGE_MAP()

void CHerdSimDlg::OnHelpContents()
{
    WinHelp(0, HELP_CONTENTS);
}

void CMainMenu::OnHelpContents()
{
    WinHelp(0, HELP_CONTENTS);
}

void CInput1::OnHelpContents()
{ 
    WinHelp(0,HELP_CONTENTS);
}

void CInput2::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

void CInput2A::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

void CInput2B::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

void CInput4::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

void YearOption::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

void CInput8::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

void CInput7::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

void CInput6::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

void CInput5::OnHelpContents()
{
    WinHelp(0,HELP_CONTENTS);
}

/////////////////////////////////////////////////////////////////////////////////////////////
// CINPUT9 dialog
CINPUT9::CINPUT9(CWnd* pParent /*=NULL*/) : CDialog(CINPUT9::IDD, pParent)
{
    // {AFX_DATA_INIT(CINPUT9)
    m_dayin1 = 0;
    m_dayin2 = 0;
    m_dayin3 = 0;
    m_dayin4 = 0;
    m_dayin5 = 0;
    m_dayin6 = 0;
    m_dayout1 = 0;
    m_dayout2 = 0;
    m_dayout3 = 0;
    m_dayout4 = 0;
    m_dayout5 = 0;
    m_dayout6 = 0;
    m_hland1 = 0;
    m_hland2 = 0;
    m_hland3 = 0;
    m_hland4 = 0;
    m_hland5 = 0;
    m_hland6 = 0;
    m_monthin1 = 0;
    m_monthin2 = 0;
    m_monthin3 = 0;
    m_monthin4 = 0;
    m_monthin5 = 0;
    m_monthin6 = 0;
    m_monthout1 = 0;
    m_monthout2 = 0;
    m_monthout3 = 0;
    m_monthout4 = 0;
    m_monthout5 = 0;
    m_monthout6 = 0;
    m_tpas1 = 0;
    m_tpas2 = 0;
    m_tpas3 = 0;
    m_tpas4 = 0;
    m_tpas5 = 0;
    m_tpas6 = 0;
    m_wland1 = 0;
    m_wland2 = 0;
    m_wland3 = 0;
    m_wland4 = 0;
    m_wland5 = 0;
    m_wland6 = 0;
    m_numpas = 0;
    m_numpasture = 0;
    }

    void CINPUT9::DoDataExchange(CDataExchange* pDX)
BEGIN_MESSAGE_MAP(CINPUT9, CDialog)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_EN_CHANGE(IDC_NUMPASTURE, OnChangeNumpasture)
END_MESSAGE_MAP()
BOOL CInput5::OnInitDialog()
{
    CDialog::OnInitDialog();
    return TRUE;
}

BOOL CInput9::OnInitDialog()
{
    CDialog::OnInitDialog();
    StartUp();
    return TRUE;
}

CInput10::CInput10(CWnd* pParent /*=NULL*/)
        : CDialog(CInput10::IDD, pParent)
{
    m_dayrin1 = 0;
    m_dayrin2 = 0;
    m_dayrin3 = 0;
    m_dayrin4 = 0;
    m_dayrin5 = 0;
    m_dayrin6 = 0;
    m_dayrout1 = 0;
    m_dayrout2 = 0;
    m_dayrout3 = 0;
    m_dayrout4 = 0;
    m_dayrout5 = 0;
    m_dayrout6 = 0;
    m_hrland1 = 0;
    m_hrland2 = 0;
    m_hrland3 = 0;
    m_hrland4 = 0;
    m_hrland5 = 0;
    m_hrland6 = 0;
    m_monthrin1 = 0;
    m_monthrin2 = 0;
    m_monthrin3 = 0;
    m_monthrin4 = 0;
    m_monthrin5 = 0;
    m_monthrin6 = 0;
    m_monthrout1 = 0;
    m_monthrout2 = 0;
    m_monthrout3 = 0;
    m_monthrout4 = 0;
    m_monthrout5 = 0;
    m_monthrout6 = 0;
    m_numrpasture = 0;
    m_trpas1 = 0;
m_trpas2 = 0;
m_trpas3 = 0;
m_trpas4 = 0;
m_trpas5 = 0;
m_trpas6 = 0;
m_wrland1 = 0;
m_wrland2 = 0;
m_wrland3 = 0;
m_wrland4 = 0;
m_wrland5 = 0;
m_wrland6 = 0;
}

void CInput10::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_DAYRIN1, m_dayrin1);
    DDX_Text(pDX, IDC_DAYRIN2, m_dayrin2);
    DDX_Text(pDX, IDC_DAYRIN3, m_dayrin3);
    DDX_Text(pDX, IDC_DAYRIN4, m_dayrin4);
    DDX_Text(pDX, IDC_DAYRIN5, m_dayrin5);
    DDX_Text(pDX, IDC_DAYRIN6, m_dayrin6);
    DDX_Text(pDX, IDC_DAYROUT1, m_dayrout1);
    DDX_Text(pDX, IDC_DAYROUT2, m_dayrout2);
    DDX_Text(pDX, IDC_DAYROUT3, m_dayrout3);
    DDX_Text(pDX, IDC_DAYROUT4, m_dayrout4);
    DDX_Text(pDX, IDC_DAYROUT5, m_dayrout5);
    DDX_Text(pDX, IDC_DAYROUT6, m_dayrout6);
    DDX_Text(pDX, IDC_HRLAND1, m_hrland1);
    DDX_Text(pDX, IDC_HRLAND2, m_hrland2);
    DDX_Text(pDX, IDC_HRLAND3, m_hrland3);
    DDX_Text(pDX, IDC_HRLAND4, m_hrland4);
    DDX_Text(pDX, IDC_HRLAND5, m_hrland5);
    DDX_Text(pDX, IDC_HRLAND6, m_hrland6);
    DDX_Text(pDX, IDC_MONTHRIN1, m_monthrin1);
    DDX_Text(pDX, IDC_MONTHRIN2, m_monthrin2);
    DDX_Text(pDX, IDC_MONTHRIN3, m_monthrin3);
    DDX_Text(pDX, IDC_MONTHRIN4, m_monthrin4);
    DDX_Text(pDX, IDC_MONTHRIN5, m_monthrin5);
    DDX_Text(pDX, IDC_MONTHRIN6, m_monthrin6);
    DDX_Text(pDX, IDC_MONTHROUT1, m_monthrout1);
    DDX_Text(pDX, IDC_MONTHROUT2, m_monthrout2);
    DDX_Text(pDX, IDC_MONTHROUT3, m_monthrout3);
    DDX_Text(pDX, IDC_MONTHROUT4, m_monthrout4);
    DDX_Text(pDX, IDC_MONTHROUT5, m_monthrout5);
    DDX_Text(pDX, IDC_MONTHROUT6, m_monthrout6);
    DDX_Text(pDX, IDC_NUMRPASTURE, m_numrpasture);
    DDX_Text(pDX, IDC_TRPAS1, m_trpas1);
    DDX_Text(pDX, IDC_TRPAS2, m_trpas2);
    DDX_Text(pDX, IDC_TRPAS3, m_trpas3);
    DDX_Text(pDX, IDC_TRPAS4, m_trpas4);
DDX_Text(pDX, IDC_TRPAS5, m_trpas5);
DDX_Text(pDX, IDC_TRPAS6, m_trpas6);
DDX_Text(pDX, IDC_WRLAND1, m_wrland1);
DDX_Text(pDX, IDC_WRLAND2, m_wrland2);
DDX_Text(pDX, IDC_WRLAND3, m_wrland3);
DDX_Text(pDX, IDC_WRLAND4, m_wrland4);
DDX_Text(pDX, IDC_WRLAND5, m_wrland5);
DDX_Text(pDX, IDC_WRLAND6, m_wrland6);
}

BEGIN_MESSAGE_MAP(CInput10, CDialog)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_EN_CHANGE(IDC_NUMRPASTURE, OnChangeNumrpastru)
END_MESSAGE_MAP()

BOOL CInput10::OnInitDialog()
{
    CDialog::OnInitDialog();
    StartUp();
    return TRUE;
}

void CINPUT9::OnNext()
{
    UpdateData();
    CDialog::EndDialog(IDC_NEXT);
}

void CInput10::OnPrev()
{
    UpdateData();
    CDialog::EndDialog(IDC_PREV);
}

void CInput10::OnNext()
{
    UpdateData();
    CDialog::EndDialog(IDC_NEXT);
}

void CINPUT9::StartUp()
{
    CEdit* tp1 = (CEdit *) GetDlgItem(IDC_TPAS1);
    CEdit* tp2 = (CEdit *) GetDlgItem(IDC_TPAS2);
    CEdit* tp3 = (CEdit *) GetDlgItem(IDC_TPAS3);
    CEdit* tp4 = (CEdit *) GetDlgItem(IDC_TPAS4);
    CEdit* tp5 = (CEdit *) GetDlgItem(IDC_TPAS5);
    CEdit* tp6 = (CEdit *) GetDlgItem(IDC_TPAS6);
    CEdit* hl1 = (CEdit *) GetDlgItem(IDC_HLAND1);
CEdit* h12 = (CEdit *) GetDlgItem(IDC_HLAND2);
CEdit* h13 = (CEdit *) GetDlgItem(IDC_HLAND3);
CEdit* h14 = (CEdit *) GetDlgItem(IDC_HLAND4);
CEdit* h15 = (CEdit *) GetDlgItem(IDC_HLAND5);
CEdit* h16 = (CEdit *) GetDlgItem(IDC_HLAND6);
CEdit* w11 = (CEdit *) GetDlgItem(IDC_WLAND1);
CEdit* w12 = (CEdit *) GetDlgItem(IDC_WLAND2);
CEdit* w13 = (CEdit *) GetDlgItem(IDC_WLAND3);
CEdit* w14 = (CEdit *) GetDlgItem(IDC_WLAND4);
CEdit* w15 = (CEdit *) GetDlgItem(IDC_WLAND5);
CEdit* w16 = (CEdit *) GetDlgItem(IDC_WLAND6);
CEdit* mi1 = (CEdit *) GetDlgItem(IDC_MONTHIN1);
CEdit* mi2 = (CEdit *) GetDlgItem(IDC_MONTHIN2);
CEdit* mi3 = (CEdit *) GetDlgItem(IDC_MONTHIN3);
CEdit* mi4 = (CEdit *) GetDlgItem(IDC_MONTHIN4);
CEdit* mi5 = (CEdit *) GetDlgItem(IDC_MONTHIN5);
CEdit* mi6 = (CEdit *) GetDlgItem(IDC_MONTHIN6);
CEdit* di1 = (CEdit *) GetDlgItem(IDC_DAYIN1);
CEdit* di2 = (CEdit *) GetDlgItem(IDC_DAYIN2);
CEdit* di3 = (CEdit *) GetDlgItem(IDC_DAYIN3);
CEdit* di4 = (CEdit *) GetDlgItem(IDC_DAYIN4);
CEdit* di5 = (CEdit *) GetDlgItem(IDC_DAYIN5);
CEdit* di6 = (CEdit *) GetDlgItem(IDC_DAYIN6);
CEdit* mo1 = (CEdit *) GetDlgItem(IDC_MONTHOUT1);
CEdit* mo2 = (CEdit *) GetDlgItem(IDC_MONTHOUT2);
CEdit* mo3 = (CEdit *) GetDlgItem(IDC_MONTHOUT3);
CEdit* mo4 = (CEdit *) GetDlgItem(IDC_MONTHOUT4);
CEdit* mo5 = (CEdit *) GetDlgItem(IDC_MONTHOUT5);
CEdit* mo6 = (CEdit *) GetDlgItem(IDC_MONTHOUT6);
CEdit* do1 = (CEdit *) GetDlgItem(IDC_DAYOUT1);
CEdit* do2 = (CEdit *) GetDlgItem(IDC_DAYOUT2);
CEdit* do3 = (CEdit *) GetDlgItem(IDC_DAYOUT3);
CEdit* do4 = (CEdit *) GetDlgItem(IDC_DAYOUT4);
CEdit* do5 = (CEdit *) GetDlgItem(IDC_DAYOUT5);
CEdit* do6 = (CEdit *) GetDlgItem(IDC_DAYOUT6);
UpdateData();
switch(m_numpasture){
case (6): tp6->EnableWindow(true); w6->EnableWindow(true);
    h6->EnableWindow(true); d6->EnableWindow(true); m6-
    >EnableWindow(true); case (5): tp5->EnableWindow(true); w5->EnableWindow(true);
    h5->EnableWindow(true); d5->EnableWindow(true); m5-
    >EnableWindow(true); case (4): tp4->EnableWindow(true); w4->EnableWindow(true);
    h4->EnableWindow(true); d4->EnableWindow(true); m4-
    >EnableWindow(true); case (3): tp3->EnableWindow(true); w3->EnableWindow(true);
    h3->EnableWindow(true);
mi3->EnableWindow(true); di3->EnableWindow(true); mo3->EnableWindow(true); do3->EnableWindow(true);
case (2): tp2->EnableWindow(true); wl2->EnableWindow(true); hl2->EnableWindow(true);
mi2->EnableWindow(true); di2->EnableWindow(true); mo2->EnableWindow(true); do2->EnableWindow(true);
case (1): tp1->EnableWindow(true); wl1->EnableWindow(true); hl1->EnableWindow(true);
mi1->EnableWindow(true); di1->EnableWindow(true); mo1->EnableWindow(true); do1->EnableWindow(true);
}
switch(m_numpasture){
case (0): tp1->EnableWindow(false); wl1->EnableWindow(false); hl1->EnableWindow(false);
mi1->EnableWindow(false); di1->EnableWindow(false); mo1->EnableWindow(false); do1->EnableWindow(false);
case (1): tp2->EnableWindow(false); hl2->EnableWindow(false);
mi2->EnableWindow(false); di2->EnableWindow(false); mo2->EnableWindow(false); do2->EnableWindow(false);
case (2): tp3->EnableWindow(false); hl3->EnableWindow(false);
mi3->EnableWindow(false); di3->EnableWindow(false); mo3->EnableWindow(false); do3->EnableWindow(false);
case (3): tp4->EnableWindow(false); hl4->EnableWindow(false);
mi4->EnableWindow(false); di4->EnableWindow(false); mo4->EnableWindow(false); do4->EnableWindow(false);
case (4): tp5->EnableWindow(false); hl5->EnableWindow(false);
mi5->EnableWindow(false); di5->EnableWindow(false); mo5->EnableWindow(false); do5->EnableWindow(false);
case (5): tp6->EnableWindow(false); hl6->EnableWindow(false);
mi6->EnableWindow(false); di6->EnableWindow(false); mo6->EnableWindow(false); do6->EnableWindow(false);
}
void CInput10::StartUp()
{
    CEdit* tp1 = (CEdit *) GetDlgItem(IDC_TRPAS1);
    CEdit* tp2 = (CEdit *) GetDlgItem(IDC_TRPAS2);
    CEdit* tp3 = (CEdit *) GetDlgItem(IDC_TRPAS3);
    CEdit* tp4 = (CEdit *) GetDlgItem(IDC_TRPAS4);
    CEdit* tp5 = (CEdit *) GetDlgItem(IDC_TRPAS5);
    CEdit* tp6 = (CEdit *) GetDlgItem(IDC_TRPAS6);
    CEdit* hl1 = (CEdit *) GetDlgItem(IDC_HRLAND1);
    CEdit* hl2 = (CEdit *) GetDlgItem(IDC_HRLAND2);
    CEdit* hl3 = (CEdit *) GetDlgItem(IDC_HRLAND3);
    CEdit* hl4 = (CEdit *) GetDlgItem(IDC_HRLAND4);
CEdit* hl5 = (CEdit*) GetDlgItem(IDC_HRLAND5);
CEdit* hl6 = (CEdit*) GetDlgItem(IDC_HRLAND6);
CEdit* w11 = (CEdit*) GetDlgItem(IDC_WRLAND1);
CEdit* w12 = (CEdit*) GetDlgItem(IDC_WRLAND2);
CEdit* w13 = (CEdit*) GetDlgItem(IDC_WRLAND3);
CEdit* w14 = (CEdit*) GetDlgItem(IDC_WRLAND4);
CEdit* w15 = (CEdit*) GetDlgItem(IDC_WRLAND5);
CEdit* w16 = (CEdit*) GetDlgItem(IDC_WRLAND6);
CEdit* m1 = (CEdit*) GetDlgItem(IDC_MONTHRIN1);
CEdit* m2 = (CEdit*) GetDlgItem(IDC_MONTHRIN2);
CEdit* m3 = (CEdit*) GetDlgItem(IDC_MONTHRIN3);
CEdit* m4 = (CEdit*) GetDlgItem(IDC_MONTHRIN4);
CEdit* m5 = (CEdit*) GetDlgItem(IDC_MONTHRIN5);
CEdit* m6 = (CEdit*) GetDlgItem(IDC_MONTHRIN6);
CEdit* d1 = (CEdit*) GetDlgItem(IDC_DAYRIN1);
CEdit* d2 = (CEdit*) GetDlgItem(IDC_DAYRIN2);
CEdit* d3 = (CEdit*) GetDlgItem(IDC_DAYRIN3);
CEdit* d4 = (CEdit*) GetDlgItem(IDC_DAYRIN4);
CEdit* d5 = (CEdit*) GetDlgItem(IDC_DAYRIN5);
CEdit* d6 = (CEdit*) GetDlgItem(IDC_DAYRIN6);
CEdit* m1 = (CEdit*) GetDlgItem(IDC_MONTHROUT1);
CEdit* m2 = (CEdit*) GetDlgItem(IDC_MONTHROUT2);
CEdit* m3 = (CEdit*) GetDlgItem(IDC_MONTHROUT3);
CEdit* m4 = (CEdit*) GetDlgItem(IDC_MONTHROUT4);
CEdit* m5 = (CEdit*) GetDlgItem(IDC_MONTHROUT5);
CEdit* m6 = (CEdit*) GetDlgItem(IDC_MONTHROUT6);
CEdit* d1 = (CEdit*) GetDlgItem(IDC_DAYROUT1);
CEdit* d2 = (CEdit*) GetDlgItem(IDC_DAYROUT2);
CEdit* d3 = (CEdit*) GetDlgItem(IDC_DAYROUT3);
CEdit* d4 = (CEdit*) GetDlgItem(IDC_DAYROUT4);
CEdit* d5 = (CEdit*) GetDlgItem(IDC_DAYROUT5);
CEdit* d6 = (CEdit*) GetDlgItem(IDC_DAYROUT6);
UpdateData();
switch(m_numrpasture){
  case (6): tp6->EnableWindow(true); w6->EnableWindow(true);
    h6->EnableWindow(true);
    m6->EnableWindow(true); d6->EnableWindow(true);
  case (5): tp5->EnableWindow(true); w5->EnableWindow(true);
    h5->EnableWindow(true);
    m5->EnableWindow(true); d5->EnableWindow(true);
  case (4): tp4->EnableWindow(true); w4->EnableWindow(true);
    h4->EnableWindow(true);
    m4->EnableWindow(true); d4->EnableWindow(true);
  case (3): tp3->EnableWindow(true); w3->EnableWindow(true);
    h3->EnableWindow(true);
    m3->EnableWindow(true); d3->EnableWindow(true);
  case (2): tp2->EnableWindow(true); w2->EnableWindow(true);

hl2->EnableWindow(true);
mi2->EnableWindow(true); di2->EnableWindow(true); mo2->EnableWindow(true); do2->EnableWindow(true);
case (1): tp1->EnableWindow(true); w11->EnableWindow(true);
hl1->EnableWindow(true);
mi1->EnableWindow(true); di1->EnableWindow(true); mo1->EnableWindow(true); do1->EnableWindow(true);
}
switch (m_numrpasture){
case (0): tp1->EnableWindow(false); w11->EnableWindow(false);
mi1->EnableWindow(false); di1->EnableWindow(false); mo1->EnableWindow(false); do1->EnableWindow(false);
case (1): tp2->EnableWindow(false); w12->EnableWindow(false);
mi2->EnableWindow(false); di2->EnableWindow(false); mo2->EnableWindow(false); do2->EnableWindow(false);
case (2): tp3->EnableWindow(false); w13->EnableWindow(false);
mi3->EnableWindow(false); di3->EnableWindow(false); mo3->EnableWindow(false); do3->EnableWindow(false);
case (3): tp4->EnableWindow(false); w14->EnableWindow(false);
mi4->EnableWindow(false); di4->EnableWindow(false); mo4->EnableWindow(false); do4->EnableWindow(false);
case (4): tp5->EnableWindow(false); w15->EnableWindow(false);
mi5->EnableWindow(false); di5->EnableWindow(false); mo5->EnableWindow(false); do5->EnableWindow(false);
case (5): tp6->EnableWindow(false); w16->EnableWindow(false);
mi6->EnableWindow(false); di6->EnableWindow(false); mo6->EnableWindow(false); do6->EnableWindow(false);
}

void CInput10::OnChangeNumrpasture()
{
StartUp();
}

void CINPUT9::OnChangeNumpasture()
{
StartUp();
}

/////////////////////////////////////////////////////////////////////////////
// CProgress dialog
CProgress::CProgress(CWnd* pParent /*=NULL*/) :
CDialog(CProgress::IDD, pParent)
{
m_caption1 = _T("")
;

void CProgress::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_PATIENT, m_caption1);
    DDV_MaxChars(pDX, m_caption1, 40);
}

BEGIN_MESSAGE_MAP(CProgress, CDialog)
END_MESSAGE_MAP()

void CProgress::onStepFunction()
{
    CProgressCtrl* tp1 = (CProgressCtrl *) GetDlgItem(IDC_PROGRESS1);
    tp1->StepIt();
}

void CProgress::onStartUp(int numOfYears)
{
    CProgressCtrl* tp1 = (CProgressCtrl *) GetDlgItem(IDC_PROGRESS1);
    tp1->SetRange(0, numOfYears * 10);
}

Cinput11::Cinput11(CWnd* pParent /*=NULL*/)
 : CDialog(Cinput11::IDD, pParent)
{
    m_adg1 = FALSE;
    m_seasonal = FALSE;
    m_antiadg = 0.0;
    m_antiadg1 = 0.0;
    m_antiadg2 = 0.0;
    m_antiadg3 = 0.0;
    m_antiadg4 = 0.0;
    m_retstocker = 0;
    //}AFX_DATA_INIT
}

void Cinput11::DoDataExchange(CDataExchange* pDX)
{ 
    CDialog::DoDataExchange(pDX);
    DDX_Check(pDX, IDC_ADG1, m_adg1);
    DDX_Check(pDX, IDC_SEASONAL, m_seasonal);
    DDX_Text(pDX, IDC_ANTIADG, m_antiadg);
    DDV_MinMaxDouble(pDX, m_antiadg, 0., 999.);
}
DDX_Text(pDX, IDC_ANTIADG1, m_antiadg1);
DDV_MinMaxDouble(pDX, pDX, m_antiadg1, 0., 999.);
DDX_Text(pDX, IDC_ANTIADG2, m_antiadg2);
DDV_MinMaxDouble(pDX, pDX, m_antiadg2, 0., 999.);
DDX_Text(pDX, IDC_ANTIADG3, m_antiadg3);
DDV_MinMaxDouble(pDX, pDX, m_antiadg3, 0., 999.);
DDX_Text(pDX, IDC_ANTIADG4, m_antiadg4);
DDV_MinMaxDouble(pDX, pDX, m_antiadg4, 0., 999.);
DDX_Text(pDX, IDC_RETSTOCKER, m_retstocker);
DDV_MinMaxInt(pDX, pDX, m_retstocker, 0, 9999);
}
BEGIN_MESSAGE_MAP(Cinput11, CDialog)
ON_BN_CLICKED(IDC_PREV, OnPrev)
ON_BN_CLICKED(IDC_NEXT, OnNext)
ON_BN_CLICKED(IDC_MENU, OnMenu)
ON_BN_CLICKED(IDC_HELP, OnHelp)
ON_BN_CLICKED(IDC_ADG1, OnAdg)
ON_BN_CLICKED(IDC_SEASONAL, OnSeasonal)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////////////
void Cinput11::OnPrev()
{
UpdateData();
CDialog::EndDialog(IDC_PREV);
}

void Cinput11::OnNext()
{
UpdateData();
CDialog::EndDialog(IDC_NEXT);
}

void Cinput11::OnMenu()
{
UpdateData();
CDialog::EndDialog(IDC_MENU);
}

void Cinput11::OnHelp()
{
WinHelp(0,HELP_CONTENTS);
}

void Cinput11::OnAdg()
{
CStatic* antiadg = (CStatic *) GetDlgItem(IDC_ADGSTATIC);
CEdit* antiadgbox = (CEdit *) GetDlgItem(IDC_ANTIADG);
CStatic* seasonalstatic = (CStatic *)GetDlgItem(IDC_SEASONAL);
}
CButton* seasonal = (CButton *)GetDlgItem(IDC_SEASONAL);
CEdit* antiadg1 = (CEdit *) GetDlgItem(IDC_ANTIADG1);
CEdit* antiadg2 = (CEdit *) GetDlgItem(IDC_ANTIADG2);
CEdit* antiadg3 = (CEdit *) GetDlgItem(IDC_ANTIADG3);
CEdit* antiadg4 = (CEdit *) GetDlgItem(IDC_ANTIADG4);
CStatic* janstatic = (CStatic *) GetDlgItem(IDC_JANSTATIC);
CStatic* aprilstatic = (CStatic *) GetDlgItem(IDC_APRILSTATIC);
CStatic* julystatic = (CStatic *) GetDlgItem(IDC_JULYSTATIC);
CStatic* octstatic = (CStatic *) GetDlgItem(IDC_OCTSTATIC);

if (!IsDlgButtonChecked(IDC_ADG1)) {
    antiadg->EnableWindow(TRUE);
    antiadgbox->EnableWindow(TRUE);
    seasonalstatic->EnableWindow(TRUE);
    seasonal->EnableWindow(TRUE);
    OnSeasonal();
} else {
    antiadg->EnableWindow(FALSE);
    antiadgbox->EnableWindow(FALSE);
    seasonalstatic->EnableWindow(FALSE);
    seasonal->EnableWindow(FALSE);
    antiadg->EnableWindow(FALSE);
    antiadgbox->EnableWindow(FALSE);
    antiadg1->EnableWindow(FALSE);
    antiadg2->EnableWindow(FALSE);
    antiadg3->EnableWindow(FALSE);
    antiadg4->EnableWindow(FALSE);
    janstatic->EnableWindow(FALSE);
    aprilstatic->EnableWindow(FALSE);
    julystatic->EnableWindow(FALSE);
    octstatic->EnableWindow(FALSE);
}

BOOL Cinput11::OnInitDialog()
{
    CDialog::OnInitDialog();
    OnAdg();
    return TRUE;
}

void Cinput11::OnSeasonal()
{
    CStatic* antiadg = (CStatic *) GetDlgItem(IDC_ADGSTATIC);
    CEdit* antiadgbox = (CEdit *) GetDlgItem(IDC_ANTIADG);
    CEdit* antiadg1 = (CEdit *) GetDlgItem(IDC_ANTIADG1);
    CEdit* antiadg2 = (CEdit *) GetDlgItem(IDC_ANTIADG2);
    CEdit* antiadg3 = (CEdit *) GetDlgItem(IDC_ANTIADG3);
    CEdit* antiadg4 = (CEdit *) GetDlgItem(IDC_ANTIADG4);
CStatic* janstatic = (CStatic *) GetDlgItem(IDC_JANSTATIC);
CStatic* aprilstatic = (CStatic *) GetDlgItem(IDC_APRILSTATIC);
CStatic* julystatic = (CStatic *) GetDlgItem(IDC_JULYSTATIC);
CStatic* octstatic = (CStatic *) GetDlgItem(IDC_OCTSTATIC);
if (IsDlgButtonChecked(IDC_SEASONAL)){
    antiadg->EnableWindow(FALSE);
    antiadgbox->EnableWindow(FALSE);
    antiadg1->EnableWindow(TRUE);
    antiadg2->EnableWindow(TRUE);
    antiadg3->EnableWindow(TRUE);
    antiadg4->EnableWindow(TRUE);
    janstatic->EnableWindow(TRUE);
    aprilstatic->EnableWindow(TRUE);
    julystatic->EnableWindow(TRUE);
    octstatic->EnableWindow(TRUE);
}
else{
    antiadg->EnableWindow(TRUE);
    antiadgbox->EnableWindow(TRUE);
    antiadg1->EnableWindow(FALSE);
    antiadg2->EnableWindow(FALSE);
    antiadg3->EnableWindow(FALSE);
    antiadg4->EnableWindow(FALSE);
    janstatic->EnableWindow(FALSE);
    aprilstatic->EnableWindow(FALSE);
    julystatic->EnableWindow(FALSE);
    octstatic->EnableWindow(FALSE);
}
}
Market.cpp
#include "stdafx.h"
#include "HerdSim.h"
#include "HerdSimDlg.h"
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include "Market.h"
#include "cow.h"
#include "Nutrit.h"
#include "tools.h"
extern char CULAG;
extern char CULPO;
int AgedCulls;
int PregCullCow;
extern int NumStkrYEnd;
extern int Sterile;
extern int Dystocia2;
extern int Dystocia3;
extern int Dystocia4;
extern int FDBS;
extern int FDBHS;
extern int FDBHS2;
extern int NumHeifers;
extern int NumCows;
extern int CyclHeifer1;
extern int CyclHeifer2;
extern int CyclHeifer3;
extern int CyclHeifer21;
extern int CyclHeifer22;
extern int CyclHeifer23;
extern int PregHef1;
extern int PregHef2;
extern int PregHef3;
extern int PregHef21;
extern int PregHef22;
extern int PregHef23;
extern int CyclCow1;
extern int CyclCow2;
extern int CyclCow3;
extern int CyclCow21;
extern int CyclCow22;
extern int CyclCow23;
extern int PregCow1;
extern int PregCow2;
extern int PregCow3;
extern int PregCow21;
extern int PregCow22;
extern int PregCow23;
extern int FDBS2;
extern int SEAS;
extern int SEAS2;
extern char SpringCalfSeasn;
extern char FallCalfSeasn;
extern int FARMSIZE;
extern int ARTI;
extern double AVGWT;
extern double AVGBWT;
extern double AVGBWTH;
extern int DRET;
int SELL;
extern int NBULLS;
extern char NIPSH;
extern char NIPSS;
extern int DNUMHS;
extern int DNUMSS;
extern int CurrentSIZE;
extern int CurrentSIZES;
extern double FeedCost;
extern double STFeedCost;
extern int currentDATE;
extern int YEAR;
double PRAC;
double PRAS;
double RESP;
double RESPS;
extern int DNUMF;
extern int DNUM;
extern int CNUMB;
extern int CONUM;
extern int COWS;
extern int BreedingCows;
extern int CulledCows;
extern int BreedingReplacement;
extern int IMPH;
extern int IMPS;
extern int NumAvailRepla;
extern int PNUM;
extern double CNRAT;
extern int ACDAT;
extern int DNUMH;
extern double CRATE;
extern double WRATE;
extern double CULRAT;
extern double AvServConcp;
extern int SELEC;
extern int DNUMS;
int BUSE;
extern CString FARM;
extern CString ADDR;
extern CString FOWN;
extern CString FMAN;
extern CString LMAN;
double BCST;
double DEADC;
int REPLR;
double NHFC1;
double NSTC1;
double TWSS1;
double TWHS1;
double TTSTR1;
double TTHFR1;
double TTSTR2;
double TTHFR2;
extern int PHEF;
extern int POHEF;
double CostofReplacements;
extern int TNCOH;
extern int TNSCOH;
double TotWeanWtH;
double TotWeanWtS;
extern double AVFWEANAGE;
extern double AVSWEANAGE;
int FSELLDATE;
int SSELLDATE;
int FSCALF;
int SSCALF;
extern int NumBreedSeasns;
extern int BCscore[10];
extern AgeBreeders[15];
extern double AvGestLen;
extern int CowsFailWean;
extern int CowsFail;
extern int CowsAbort;
extern int CowsEelig;
extern int EligCowsOpen;
extern int CowsInEelig;
extern int CowsEelig1;
extern int CowsInEelig1;
extern int NumCowsYend;
extern int TOTALAGE;
extern double hayConsump;
extern double Concentrate;
extern double hayConsumpS;
extern double ConcentrateS;
extern double TTGRO;
extern double TPDMIFA;
extern double TSPDMIFA;
extern double hayAVAIL;
extern int calcuADG;
extern double antiADG;
extern double TotAnnSen;
extern double CarryOver;
double THFR1;
double TSTR1;
double TBDS1;
double ASHKH1;
double ASHKS1;
double YDCA1;
double TBDH1;
double TDSPH1;
double TDSPS1;
double TESPH1;
double TESP11;
double TESPS1;
double TFEC11;
double TTSTR0;
double TTHF10;
double ATTSTR0;
double ATTHF10;
extern int WEANFDATE;
extern int WEANSDATE;

Market::Market() {
    AgedCulls=0;
PregCullCow=0;
    AICS=25.0;
    BASS=3.00;
    HEDG1=0;
    HEDG2=0;
    BCST=1200.00;
    BFRC=7.00;
    BFRS=3.00;
    BSAL=0.2;
    DMandSC=0.0013006;
    DPREM=0.05;
    EPREM=0.065;
    FICA=0.0765;
    HDEX=80.00;
    HOUR=8.0;
    HOURS=3.5;
    INSURS=5.50;
    INTR=9.5;
    INTRS=9.5;
    IPCS=1.80;
    LORT=0.80;
    LORTS=0.80;
    LSCS=2.00;
    LSCSS=3.00;
    MCCO=10.00;
    MCCOS=10.00;
    PMNA=12.97;
    PMNS=12.97;
    PRAC=20.00;
    PRAS=20.00;
    RESP=100.00;
RESPS=100.00;
SALT=68.00;
SMCS=8.00;
SMCSS=8.00;
TCCS=5.20;
TCSC=4.50;
TCSCS=4.50;
UTLC=1.25;
UTLCH=1.75;
VMCS=12.25;
VMCSS=5.63;
WAGE=8.00;
BUSE=3;
DIST=40;
ONIT=0;
SALC=1;
SALH=2;
SALO=1;
SALS=2;
SALSH=2;
SALSS=2;
PRSA1=72.50;
PRSA2=70.00;
PRSA3=71.00;
SPRSA1=70.00;
SPRSA2=68.00;
SPRSA3=70.50;
CPRSA1=36.00;
CPRSA2=40.00;
CPRSA3=41.00;
CCRV=0;
PRIS=0;
TCCR=0;
NCCS=0;
TWCC=0;
TBDOC=0;
TDSPOC=0;
OCRV=0;
TOCR=0;
TWOS=0;
NOC=0;
ASHKOC=0;
TSHKOC=0;
THFR=0;
NHFC=0;
NSTC=0;
NHFC1=0;
NSTC1=0;
TWSS=0;
TWHS=0;
TWSS1=0;
TWHS1=0;
TTSTR1 = 0;
TTHFR1 = 0;
TTSTR2 = 0;
TTHFR2 = 0;
TBDH=0;
TDSPH=0;
TDSPS=0;
TESPH=0;
TESPS=0;
TFECA=0;
TSHKH=0;
TSHKS=0;
TSTR=0;
TBDS=0;
ASHKH=0;
ASHKS=0;
YDCA=0;
HFRV=0;
STRV=0;
NSHFC=0;
NSSTC=0;
TWSHS=0;
TWSSS=0;
TBDSH=0;
TBDSS=0;
TDSPSH=0;
TDSPSS=0;
TESPSS=0;
TESPSH=0;
TFECAS=0;
TotalPurchaseCost=0;
CostofReplacements=0;
TotWeanWtH=0;
TotWeanWtS=0;
TSHFR=0;
TSHKSH=0;
TSHKSS=0;
TSSTR=0;
ASHKSH=0;
ASHKSS=0;
YDCAS=0;
FSELLDATE=0;
SELLDATE=0;
FSCALF=0;
void Market::UseInput3()
{
    CInput3* input = new CInput3;
    input->m_orphan = SALO;
    input->m_culled = SALC;
    input->m_diff = BASS;
    input->m_heifer = SALH;
    input->m_sheifer = SALSH;
    input->m_ssteer = SALSS;
    input->m_steer = SALS;
    input->m_year1 = PRSA1;
    input->m_year2 = PRSA2;
    input->m_year3 = PRSA3;
    input->m_syear1 = SPRSA1;
    input->m_syear2 = SPRSA2;
    input->m_syear3 = SPRSA3;
    input->m_cyear1 = CPRSA1;
    input->m_cyear2 = CPRSA2;
    input->m_cyear3 = CPRSA3;
    input->m_hedg1 = HEDG1;
    input->m_hedg2 = HEDG2;
    input->m_onite = ONIT;
    input->m_dist = DIST;
    int response = input->DoModal();
    if (response == IDCANCEL) {
        delete input;
        return;
    }
    SALO = input->m_orphan;
    SALC = input->m_culled;
    BASS = input->m_diff;
    SALH = input->m_heifer;
    SALSH = input->m_sheifer;
SALSS=input->m_ssteer;
SALS=input->m_steer;
PRSA1=input->m_year1;
PRSA2=input->m_year2;
PRSA3=input->m_year3;
SPRSA1=input->m_syear1;
SPRSA2=input->m_syear2;
SPRSA3=input->m_syear3;
CPRSA1=input->m_cyear1;
CPRSA2=input->m_cyear2;
CPRSA3=input->m_cyear3;
HEDG1=input->m_hedg1;
HEDG2=input->m_hedg2;
ONIT=input->m_onite;
DIST=input->m_dist;
if (response==IDC_NEXT){
    UseInput4();
}
delete input;
return;
}

void Market::UseInput4(){
CInput4* input = new CInput4;
inpu->m_aicharge=AICS;
inpu->m_calftrans=TCSC;
inpu->m_fence=BFRC;
inpu->m_implants=IPCS;
inpu->m_insurance=INSURS;
inpu->m_intr =INTR;
inpu->m_intrs =INTRS;
inpu->m_labor=HOUR;
inpu->m_laborcharge=WAGE;
inpu->m_machine=MCCO;
inpu->m_maint=PMNA;
inpu->m_mincosts=SMCS;
inpu->m_fence=BFRS;
inpu->m_slabor=HOURS;
inpu->m_smachine=MCCOS;
inpu->m_smaint=PMNS;
inpu->m_smincost=SMCS;
inpu->m_supplies=LSCSS;
inpu->m_strans=TCSCS;
inpu->m_supplies=LSCS;
inpu->m_sutils=UTLCH;
inpu->m_svet=VMCSS;
inpu->m_taxrate=FICA;
inpu->m_trans=TCCS;
inpu->m_utils=UTLC;
inpu->m_vet=VMCS;
int response = input->DoModal();
if (response==IDCANCEL){
delete input;
return;
}
AICS=input->m_aicharge;
TCSC=input->m_calftrans;
BFRC=input->m_fence;
IPCS=input->m_implants;
INSURS=input->m_insurance;
INTR=input->m_intr;
INTRS=input->m_intrs;
HOUR=input->m_labor;
WAGE=input->m_laborcharge;
MCCO=input->m_machine;
PMNA=input->m_maint;
SMCS=input->m_mincosts;
BFRS=input->m_sfence;
HOURS=input->m_slabor;
MCCOS=input->m_smachine;
PMNS=input->m_smaint;
SMCS=input->m_smincost;
LSCSS=input->m_supplies;
TCSCS=input->m_strans;
LSCS=input->m_supplies;
UTLCH=input->m_sutils;
VMCSS=input->m_svet;
FICA=input->m_taxrate;
TCCS=input->m_trans;
UTLC=input->m_utils;
VMCS=input->m_vet;
delete input;
if (response==IDC_PREV){
UseInput3();
}
return;
}

void Market::Init(){
UseInput3();
TRACE("SALSS: %i\n", SALSS);
}

int Market::SellCullCows(AnimalRecord *animal){
double PregAdjust=.876;
double DPREM = 0.05;
double FEEC = 5.00;
double PRECC = 0.00565;
double PRECCC = 0.0136;
double PRIU = 0.891;
double SHKCP = 0.04;
double YRCD = 0.03;
double YRDC = 0;
double BDCC = 0;
double SCC = 0;
double SHKC = 0;
double CPRS1[13] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
double CPRS2[13] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
double CPRS3[13] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
double SPA[13] = {0, 1.015, 1.045, 1.054, 1.041, 1.019, 1.011, 0.997, 1.009, 0.968, 0.948, 0.942, 0.950};
double PRH[11] = {0, 0, 0, 0, 0.89772, 0.90813, 0.90537, 0.88218, 0.86462, 0.8352, 0.7998};
double PRS[11] = {0, 0, 0, 0, 1.107, 1.064, 1.052, 1.000, 0.97189, 0.9398, 0.9000};
CPRS1[1] = (CPRSA1 * SPA[1]);
CPRS1[2] = (CPRSA1 * SPA[2]);
CPRS1[3] = (CPRSA1 * SPA[3]);
CPRS1[4] = (CPRSA1 * SPA[4]);
CPRS1[5] = (CPRSA1 * SPA[5]);
CPRS1[6] = (CPRSA1 * SPA[6]);
CPRS1[7] = (CPRSA1 * SPA[7]);
CPRS1[8] = (CPRSA1 * SPA[8]);
CPRS1[9] = (CPRSA1 * SPA[9]);
CPRS1[10] = (CPRSA1 * SPA[10]);
CPRS1[12] = (CPRSA1 * SPA[12]);
CPRS2[1] = (CPRSA2 * SPA[1]);
CPRS2[2] = (CPRSA2 * SPA[2]);
CPRS2[3] = (CPRSA2 * SPA[3]);
CPRS2[4] = (CPRSA2 * SPA[4]);
CPRS2[5] = (CPRSA2 * SPA[5]);
CPRS2[6] = (CPRSA2 * SPA[6]);
CPRS2[7] = (CPRSA2 * SPA[7]);
CPRS2[8] = (CPRSA2 * SPA[8]);
CPRS2[9] = (CPRSA2 * SPA[9]);
CPRS2[10] = (CPRSA2 * SPA[10]);
CPRS2[12] = (CPRSA2 * SPA[12]);
CPRS3[1] = (CPRSA3 * SPA[1]);
CPRS3[2] = (CPRSA3 * SPA[2]);
CPRS3[3] = (CPRSA3 * SPA[3]);
CPRS3[4] = (CPRSA3 * SPA[4]);
CPRS3[5] = (CPRSA3 * SPA[5]);
CPRS3[6] = (CPRSA3 * SPA[6]);
CPRS3[7] = (CPRSA3 * SPA[7]);
CPRS3[8] = (CPRSA3 * SPA[8]);
CPRS3[9] = (CPRSA3 * SPA[9]);
CPRS3[10] = (CPRSA3 * SPA[10]);
CPRS3[12]=(CPRSA3 * SPA[12]);
if (YEAR==1){
if ( (currentDATE>=1) && (currentDATE<32)){
PRIS=CPRS1[1];
}
else if ( (currentDATE>=32) && (currentDATE<60)) {
PRIS=CPRS1[2];
}
else if ( (currentDATE>=60) && (currentDATE<91)) {
PRIS=CPRS1[3];
}
else if ( (currentDATE>=91) && (currentDATE<121)) {
PRIS=CPRS1[4];
}
else if ( (currentDATE>=121) && (currentDATE<152)) {
PRIS=CPRS1[5];
}
else if ( (currentDATE>=152) && (currentDATE<182)) {
PRIS=CPRS1[6];
}
else if ( (currentDATE>=182) && (currentDATE<213)) {
PRIS=CPRS1[7];
}
else if ( (currentDATE>=213) && (currentDATE<244)) {
PRIS=CPRS1[8];
}
else if ( (currentDATE>=244) && (currentDATE<274)) {
PRIS=CPRS1[9];
}
else if ( (currentDATE>=274) && (currentDATE<305)) {
PRIS=CPRS1[10];
}
else if ( (currentDATE>=305) && (currentDATE<335)) {
PRIS=CPRS1[11];
}
else if ( (currentDATE>=335) && (currentDATE<=365)) {
PRIS=CPRS1[12];
}
}
else if (YEAR==2){
if ( (currentDATE>=1) && (currentDATE<32)) {
PRIS=CPRS2[1];
}
else if ( (currentDATE>=32) && (currentDATE<60)) {
PRIS=CPRS2[2];
}
else if ( (currentDATE>=60) && (currentDATE<91)) {
PRIS=CPRS2[3];
}
else if ( (currentDATE>=91) && (currentDATE<121)) {
PRIS=CPRS2[4];
}
else if ( (currentDATE>=121) && (currentDATE<152)) {
   PRIS=CPRS2[5];
}
else if ( (currentDATE>=152) && (currentDATE<182)) {
   PRIS=CPRS2[6];
}
else if ( (currentDATE>=182) && (currentDATE<213)) {
   PRIS=CPRS2[7];
}
else if ( (currentDATE>=213) && (currentDATE<244)) {
   PRIS=CPRS2[8];
}
else if ( (currentDATE>=244) && (currentDATE<274)) {
   PRIS=CPRS2[9];
}
else if ( (currentDATE>=274) && (currentDATE<305)) {
   PRIS=CPRS2[10];
}
else if ( (currentDATE>=305) && (currentDATE<335)) {
   PRIS=CPRS2[11];
}
else if ( (currentDATE>=335) && (currentDATE<=365)) {
   PRIS=CPRS2[12];
}
else if (YEAR==3){
   if ( (currentDATE>=1) && (currentDATE<32)) {
      PRIS=CPRS3[1];
   }
   else if ( (currentDATE>=32) && (currentDATE<60)) {
      PRIS=CPRS3[2];
   }
   else if ( (currentDATE>=60) && (currentDATE<91)) {
      PRIS=CPRS3[3];
   }
   else if ( (currentDATE>=91) && (currentDATE<121)) {
      PRIS=CPRS3[4];
   }
   else if ( (currentDATE>=121) && (currentDATE<152)) {
      PRIS=CPRS3[5];
   }
   else if ( (currentDATE>=152) && (currentDATE<182)) {
      PRIS=CPRS3[6];
   }
   else if ( (currentDATE>=182) && (currentDATE<213)) {
      PRIS=CPRS3[7];
   }
   else if ( (currentDATE>=213) && (currentDATE<244)) {
      PRIS=CPRS3[8];
   }
else if ((currentDATE>=244) && (currentDATE<274)){
    PRIS=CPRS3[9];
}
else if ((currentDATE>=274) && (currentDATE<305)){
    PRIS=CPRS3[10];
}
else if ((currentDATE>=305) && (currentDATE<335)){
    PRIS=CPRS3[11];
}
else if ((currentDATE>=335) && (currentDATE<=365)){
    PRIS=CPRS3[12];
}
if ( (animal->SEX == 3) && (animal->CULL == 1) && (animal->LACT==0) && (animal->PREG==0) ){
    if (animal->AGE < 1825){
        CCRV = ( (PRIS ) / 100) * animal->CDWT;
        TCCR += CCRV;
        NCCS++;
        TWCC += animal->CDWT;
    }
    else if (animal->AGE > 1825){
        CCRV = ( (PRIS * PRIU) / 100) * animal->CDWT;
        TCCR += CCRV;
        NCCS++;
        TWCC += animal->CDWT;
    }
    if (animal->BREED == 1){
        BDCC = CCRV * PRECC;
        TBDCC += BDCC;
    }
    else if ( (animal->BREED > 1) && (animal->COLR == 0) && (animal->PURE != 0)){
        BDCC = CCRV * PRECCC;
        TBDCC += BDCC;
    }
    else if ( (animal->BREED > 5) && (animal->COLR == 0) && (animal->PURE == 0)){
        BDCC = CCRV * PRECCC;
        TBDCC += BDCC;
    }
    if (SALC == 0){
        SCC = CCRV * DPREM;
        TDSPC += SCC;
        SHKC = SHKCP;
        TSHKC += SHKC;
        ASHKC = TSHKC / NCCS;
    }
    else if ( (SALC >= 1) && (DIST <= 50) && (ONIT == 0) ){
        SHKC = 0.022;
TSHKC += SHKC;
ASHKC = TSHKC / NCCS;
TFECAC += FEEC;
YRDC = CCRV * YRCD;
YDCAC += YRDC;
}
else if ( (SALC >= 1) && (DIST <= 50) && (ONIT == 1) ){
    SHKC = 0.042;
    TSHKC += SHKC;
    ASHKC = TSHKC / NCCS;
    TFECAC += FEEC;
    YRDC = CCRV * YRCD;
    YDCAC += YRDC;
}
else if ( (SALC >= 1) && (DIST > 50) && (ONIT == 0) ){
    SHKC = 0.022 + (.00308 * (DIST / 50) );
    TSHKC += SHKC;
    ASHKC = TSHKC / NCCS;
    TFECAC += FEEC;
    YRDC = CCRV * YRCD;
    YDCAC += YRDC;
}
else if ( (SALC >= 1) && (DIST > 50) && (ONIT == 1) ){
    SHKC = 0.042 + (.00308 * (DIST / 50) ) ;
    TSHKC += SHKC;
    ASHKC = TSHKC / NCCS;
    TFECAC += FEEC;
    YRDC = CCRV * YRCD;
    YDCAC += YRDC;
}
if ( (animal->SEX == 3) && (animal->CULL == 1) && (animal->LACT==0) && (animal->PREG==1) ){
    if (animal->AGE < 1825){
        CCRV = ((PRIU + (PregAdjust * (animal->DPREG/30))/100) * animal->CDWT) ;
        TCCR += CCRV;
        NCCS++;
        TWCC += animal->CDWT;
        PregCullCow++;
    }
    else if (animal->AGE >= 1825){
        CCRV = (((PRIU * PRIU) + (PregAdjust * (animal->DPREG/30))/100) * animal->CDWT) ;
        TCCR += CCRV;
        NCCS++;
        TWCC += animal->CDWT;
        PregCullCow++;
    }
    if (animal->AGE >= CULAG *365){
        AgedCulls++;
    }
}
if (animal->BREED == 1) {
    BDCC = CCRV * PRECC;
    TBDCC += BDCC;
} else if ((animal->BREED > 1) && (animal->COLR == 0)) {
    BDCC = CCRV * PRECCC;
    TBDCC += BDCC;
} else if (SALC == 0) {
    SCC = CCRV * DPREM;
    TDSPC += SCC;
    SHKC = SHKCP;
    TSHKC += SHKC;
    ASHKC = TSHKC / NCCS;
} else if (SALC >= 1 && (DIST <= 50) && (ONIT == 0)) {
    SHKC = 0.022;
    TSHKC += SHKC;
    ASHKC = TSHKC / NCCS;
    TFECAC += FEEC;
    YRDC = CCRV * YRCD;
    YDCAC += YRDC;
} else if (SALC >= 1 && (DIST <= 50) && (ONIT == 1)) {
    SHKC = 0.042;
    TSHKC += SHKC;
    ASHKC = TSHKC / NCCS;
    TFECAC += FEEC;
    YRDC = CCRV * YRCD;
    YDCAC += YRDC;
} else if (SALC >= 1 && (DIST > 50) && (ONIT == 0)) {
    SHKC = 0.022 + (.00308 * (DIST / 50));
    TSHKC += SHKC;
    ASHKC = TSHKC / NCCS;
    TFECAC += FEEC;
    YRDC = CCRV * YRCD;
    YDCAC += YRDC;
} else if (SALC >= 1 && (DIST > 50) && (ONIT == 1)) {
    SHKC = 0.042 + (.00308 * (DIST / 50));
    TSHKC += SHKC;
    ASHKC = TSHKC / NCCS;
    TFECAC += FEEC;
    YRDC = CCRV * YRCD;
    YDCAC += YRDC;
} if (animal->SEX==3 && animal->CULL == 1 && animal->LACT==0 && animal->PREG==0) {
BreedingCows--; 
CulledCows--; 
return (animal->ID); 
}
else if ( (animal->SEX==3) && (animal->CULL == 1) && (animal->LACT==0) && (animal->PREG==1) )
{
    BreedingCows--; 
    CulledCows--; 
    return (animal->ID); 
}
return -1;

int Market::SellOrphans(AnimalRecord *animal){
    double FEE=4.00;
    double PREH=.0042719;
    double PREHC=.014845;
    double PRES=.021527;
    double PRESC=.016717;
    double SHKOP=.03;
    double YRCD=.03;
    double BDOC=0;
    double SOC=0;
    double SHKOC=0;
    double FECA=0;
    double YRD=0;
    double DPREM=.05;
    double PRS1[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double PRS2[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double PRS3[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double SPA[13]={0,1.015,1.045,1.054,1.041,1.019,1.011,0.997,1.009,0.968,0.948,0.942,0.950};
    double PRH[11]={0,0,0,0,0,0.89772,0.90813,0.90537,0.88218,0.86462,0.8352,0.7998};
    double PRS[11]={0,0,0,0,0,1.107,1.064,1.052,1.000,0.97189,0.9398,0.9000};
    PRS1[1]= (PRSA1 * SPA[1]);
    PRS1[2]= (PRSA1 * SPA[2]);
    PRS1[3]= (PRSA1 * SPA[3]);
    PRS1[4]= (PRSA1 * SPA[4]);
    PRS1[5]= (PRSA1 * SPA[5]);
    PRS1[6]= (PRSA1 * SPA[6]);
    PRS1[7]= (PRSA1 * SPA[7]);
    PRS1[8]= (PRSA1 * SPA[8]);
    PRS1[9]= (PRSA1 * SPA[9]);
    PRS1[10]= (PRSA1 * SPA[10]);
    PRS1[12]= (PRSA1 * SPA[12]);
PRS2[1] = (PRSA2 * SPA[1]);
PRS2[2] = (PRSA2 * SPA[2]);
PRS2[3] = (PRSA2 * SPA[3]);
PRS2[4] = (PRSA2 * SPA[4]);
PRS2[5] = (PRSA2 * SPA[5]);
PRS2[6] = (PRSA2 * SPA[6]);
PRS2[7] = (PRSA2 * SPA[7]);
PRS2[8] = (PRSA2 * SPA[8]);
PRS2[9] = (PRSA2 * SPA[9]);
PRS2[10] = (PRSA2 * SPA[10]);
PRS2[12] = (PRSA2 * SPA[12]);
PRS3[1] = (PRSA3 * SPA[1]);
PRS3[2] = (PRSA3 * SPA[2]);
PRS3[3] = (PRSA3 * SPA[3]);
PRS3[4] = (PRSA3 * SPA[4]);
PRS3[5] = (PRSA3 * SPA[5]);
PRS3[6] = (PRSA3 * SPA[6]);
PRS3[7] = (PRSA3 * SPA[7]);
PRS3[8] = (PRSA3 * SPA[8]);
PRS3[9] = (PRSA3 * SPA[9]);
PRS3[10] = (PRSA3 * SPA[10]);
PRS3[12] = (PRSA3 * SPA[12]);
if (YEAR==1) {
  if ((currentDATE>=1) && (currentDATE<32)) {
    PRIS=PRS1[1];
  }
  else if ((currentDATE>=32) && (currentDATE<60)) {
    PRIS=PRS1[2];
  }
  else if ((currentDATE>=60) && (currentDATE<91)) {
    PRIS=PRS1[3];
  }
  else if ((currentDATE>=91) && (currentDATE<121)) {
    PRIS=PRS1[4];
  }
  else if ((currentDATE>=121) && (currentDATE<152)) {
    PRIS=PRS1[5];
  }
  else if ((currentDATE>=152) && (currentDATE<182)) {
    PRIS=PRS1[6];
  }
  else if ((currentDATE>=182) && (currentDATE<213)) {
    PRIS=PRS1[7];
  }
  else if ((currentDATE>=213) && (currentDATE<244)) {
    PRIS=PRS1[8];
  }
  else if ((currentDATE>=244) && (currentDATE<274)) {
    PRIS=PRS1[9];
  }
} else if ((currentDATE>=274) && (currentDATE<305)){
    PRIS=PRS1[10];
}
else if ((currentDATE>=305) && (currentDATE<335)){
    PRIS=PRS1[11];
}
else if ((currentDATE>=335) && (currentDATE<=365)){
    PRIS=PRS1[12];
}
else if (YEAR==2){
    if ((currentDATE>=1) && (currentDATE<32)){
        PRIS=PRS2[1];
    }
    else if ((currentDATE>=32) && (currentDATE<60)){
        PRIS=PRS2[2];
    }
    else if ((currentDATE>=60) && (currentDATE<91)){
        PRIS=PRS2[3];
    }
    else if ((currentDATE>=91) && (currentDATE<121)){
        PRIS=PRS2[4];
    }
    else if ((currentDATE>=121) && (currentDATE<152)){
        PRIS=PRS2[5];
    }
    else if ((currentDATE>=152) && (currentDATE<182)){
        PRIS=PRS2[6];
    }
    else if ((currentDATE>=182) && (currentDATE<213)){
        PRIS=PRS2[7];
    }
    else if ((currentDATE>=213) && (currentDATE<244)){
        PRIS=PRS2[8];
    }
    else if ((currentDATE>=244) && (currentDATE<274)){
        PRIS=PRS2[9];
    }
    else if ((currentDATE>=274) && (currentDATE<305)){
        PRIS=PRS2[10];
    }
    else if ((currentDATE>=305) && (currentDATE<335)){
        PRIS=PRS2[11];
    }
    else if ((currentDATE>=335) && (currentDATE<=365)){
        PRIS=PRS2[12];
    }
}
else if (YEAR==3){
    if ((currentDATE>=1) && (currentDATE<32)){
        PRIS=PRS2[1];
    }
}
PRIS=PRS3[1];

} else if ((currentDATE>=32) && (currentDATE<60)){
PRIS=PRS3[2];
}
else if ((currentDATE>=60) && (currentDATE<91)){
PRIS=PRS3[3];
}
else if ((currentDATE>=91) && (currentDATE<121)){
PRIS=PRS3[4];
}
else if ((currentDATE>=121) && (currentDATE<152)){
PRIS=PRS3[5];
}
else if ((currentDATE>=152) && (currentDATE<182)){
PRIS=PRS3[6];
}
else if ((currentDATE>=182) && (currentDATE<213)){
PRIS=PRS3[7];
}
else if ((currentDATE>=213) && (currentDATE<244)){
PRIS=PRS3[8];
}
else if ((currentDATE>=244) && (currentDATE<274)){
PRIS=PRS3[9];
}
else if ((currentDATE>=274) && (currentDATE<305)){
PRIS=PRS3[10];
}
else if ((currentDATE>=305) && (currentDATE<335)){
PRIS=PRS3[11];
}
else if ((currentDATE>=335) && (currentDATE<=365)){
PRIS=PRS3[12];
}
}

if ((animal->SEX==0) && (animal->SellCalf==1) && (animal->REPLA==0)){
if (animal->CDWT < 400){
OCRV=(PRIS*PRH[4])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->CDWT >= 400) && (animal->CDWT < 500)){
OCRV=(PRIS*PRH[5])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->CDWT >= 500) && (animal->CDWT < 600)){
}
OCRV=(PRIS*PRH[6])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->CDWT >= 600) && (animal->CDWT < 700)){
OCRV=(PRIS*PRH[7])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->CDWT >= 700) && (animal->CDWT < 800)){
OCRV=(PRIS*PRH[8])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->SEX==0) && (animal->SellCalf==1) &&
(animal->REPLA==1)){
if (animal->CDWT < 400){
OCRV=(PRIS*PRH[4])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->CDWT >= 400) && (animal->CDWT < 500)){
OCRV=(PRIS*PRH[5])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->CDWT >= 500) && (animal->CDWT < 600)){
OCRV=(PRIS*PRH[6])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->CDWT >= 600) && (animal->CDWT < 700)){
OCRV=(PRIS*PRH[7])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->CDWT >= 700) && (animal->CDWT < 800)){
OCRV=(PRIS*PRH[8])/100 * animal->CDWT;
TOCR+=OCRV;
NOC++;
TWOS+=animal->CDWT;
}
else if ((animal->SEX == 2) && (animal->SellCalf == 1)) {
    if (animal->CDWT < 400) {
        OCRV = ((PRIS*PRS[4])/100)*animal->CDWT;
        TOCR += OCRV;
        NOC++;
        TWOS += animal->CDWT;
    }
    else if ((animal->CDWT >= 400) && (animal->CDWT < 500)) {
        OCRV = ((PRIS*PRS[5])/100)*animal->CDWT;
        TOCR += OCRV;
        NOC++;
        TWOS += animal->CDWT;
    }
    else if ((animal->CDWT >= 500) && (animal->CDWT < 600)) {
        OCRV = ((PRIS*PRS[6])/100)*animal->CDWT;
        TOCR += OCRV;
        NOC++;
        TWOS += animal->CDWT;
    }
    else if ((animal->CDWT >= 600) && (animal->CDWT < 700)) {
        OCRV = ((PRIS*PRS[7])/100)*animal->CDWT;
        TOCR += OCRV;
        NOC++;
        TWOS += animal->CDWT;
    }
    else if ((animal->CDWT >= 700) && (animal->CDWT < 800)) {
        OCRV = ((PRIS*PRS[8])/100)*animal->CDWT;
        TOCR += OCRV;
        NOC++;
        TWOS += animal->CDWT;
    }
}
else if ((animal->SEX==0) && (animal->BREED == 1) && (animal->SellCalf==1)) {
    BDOC = OCRV * PREH;
    TBDOC += BDOC;
}
else if ((animal->SEX==0) && (animal->BREED > 1) && (animal->COLR==0) && (animal->SellCalf==1) && (animal->PURE != 0)) {
    BDOC = OCRV * PREHC;
    TBDOC += BDOC;
}
else if ((animal->SEX==0) && (animal->BREED > 5) && (animal->COLR==0) && (animal->SellCalf==1) && (animal->PURE == 0)) {
    BDOC = OCRV * PREHC;
    TBDOC += BDOC;
}
else if ((animal->SEX==2) && (animal->BREED == 1) && (animal->SellCalf==1)) {
    BDOC = OCRV * PRES;
    TBDOC += BDOC;
}
else if ((animal->SEX==2) && (animal->BREED > 1) && (animal->COLR==0) && (animal->SellCalf==1) && (animal->PURE != 0)){
    BDOC=OCRV * PRES;
    TBDOC+=BDOC;
}
else if ((animal->SEX==2) && (animal->BREED > 5) && (animal->COLR==0) && (animal->SellCalf==1) && (animal->PURE == 0)){
    BDOC=OCRV * PRES;
    TBDOC+=BDOC;
}
if ( (SALO==0) && (animal->SellCalf==1) ){  
   SOC=OCRV*DPREM;
   TDSPOC+=SOC;
   SHKOC=SHKOP;
   TSHKOC+=SHKOC;
   ASHKOC=ASHKOC/NOC;
}
else if ((SALO ==1) && (DIST <= 50) && (ONIT==0) && (animal->SellCalf==1) ){  
   SHKOC=.022;
   TSHKOC+=SHKOC;
   ASHKOC=ASHKOC/NOC;
   FECA=FEE;
   YRD=OCRV*YRCD;
   YDCA+=YRD;
   TFECA+=FECA;
}
else if ((SALO =1) && (DIST <= 50) && (ONIT==1) && (animal->SellCalf==1) ){  
   SHKOC=.042;
   TSHKOC+=SHKOC;
   ASHKOC=ASHKOC/NOC;
   FECA=FEE;
   YRD=OCRV*YRCD;
   YDCA+=YRD;
   TFECA+=FECA;
}
else if ((SALH ==1) && (DIST > 50) && (ONIT==0) && (animal->SellCalf==1) ){  
   SHKOC=.022 + (.00308 * (DIST/50));
   TSHKOC+=SHKOC;
   ASHKOC=ASHKOC/NOC;
   FECA=FEE;
   YRD=OCRV*YRCD;
   YDCA+=YRD;
   TFECA+=FECA;
}
else if ((SALH ==1) && (DIST > 50) && (ONIT==1) && (animal->SellCalf==1) ){  
   SHKOC=.042 + (.00308 * (DIST/50));
}
TSHKOC+=SHKOC;
ASHKOC=TSHKOC/NOC;
FECA=FEE;
YRD=OCRV*YRCD;
YDCA+=YRD;
TFECA+=FECA;
if (animal->SellCalf==1){
    return (animal->ID);
}
return -1;

void Market::SellWeanlings(AnimalRecord *animal){
    double FEE=4.50;
    double FEEA=1.00;
    double GFEE=.75;
    double LASS=.25;
    double PREH=.0042719;
    double PREHC=.014845;
    double PRES=.021527;
    double PRESC=.016717;
    double SHKHP=.03;
    double SHKSP=.03;
    double TELO=1.5;
    double VACA=1.00;
    double YRCD=.03;
    double BDH=0;
    double BDS=0;
    double SHFC=0;
    double SHFE=0;
    double SHKH=0;
    double FECA=0;
    double YRD=0;
    double SSC=0;
    double SHKS=0;
    double SSE=0;
    double AGFEE=.50;
    double DPREM=.05;
    double EPREM=.065;
    double PRS1[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double PRS2[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double PRS3[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double SPA[13]={0,1.015,1.045,1.054,1.041,1.019,1.011,0.997,1.009,0.968,0.948,0.942,0.950};
    double PRH[11]={0,0,0,0,0.89772,0.90813,0.90537,0.88218,0.86462,0.8352,0.7998};
    double PRS[11]={0,0,0,0,0.107,1.064,1.052,1.000,0.97189,0.9398,0.90
PRS1[1]= (PRSA1 * SPA[1]);
PRS1[2]= (PRSA1 * SPA[2]);
PRS1[3]= (PRSA1 * SPA[3]);
PRS1[4]= (PRSA1 * SPA[4]);
PRS1[5]= (PRSA1 * SPA[5]);
PRS1[6]= (PRSA1 * SPA[6]);
PRS1[7]= (PRSA1 * SPA[7]);
PRS1[8]= (PRSA1 * SPA[8]);
PRS1[9]= (PRSA1 * SPA[9]);
PRS1[10]= (PRSA1 * SPA[10]);
PRS1[12]= (PRSA1 * SPA[12]);
PRS2[1]= (PRSA2 * SPA[1]);
PRS2[2]= (PRSA2 * SPA[2]);
PRS2[3]= (PRSA2 * SPA[3]);
PRS2[4]= (PRSA2 * SPA[4]);
PRS2[5]= (PRSA2 * SPA[5]);
PRS2[6]= (PRSA2 * SPA[6]);
PRS2[7]= (PRSA2 * SPA[7]);
PRS2[8]= (PRSA2 * SPA[8]);
PRS2[9]= (PRSA2 * SPA[9]);
PRS2[10]= (PRSA2 * SPA[10]);
PRS2[12]= (PRSA2 * SPA[12]);
PRS3[1]= (PRSA3 * SPA[1]);
PRS3[2]= (PRSA3 * SPA[2]);
PRS3[3]= (PRSA3 * SPA[3]);
PRS3[4]= (PRSA3 * SPA[4]);
PRS3[5]= (PRSA3 * SPA[5]);
PRS3[6]= (PRSA3 * SPA[6]);
PRS3[7]= (PRSA3 * SPA[7]);
PRS3[8]= (PRSA3 * SPA[8]);
PRS3[9]= (PRSA3 * SPA[9]);
PRS3[10]= (PRSA3 * SPA[10]);
PRS3[12]= (PRSA3 * SPA[12]);
if ((YEAR==1) || (YEAR==0)){
  if ((currentDATE>=1) && (currentDATE<32)){
    PRIS=PRS1[1];
  }
  else if ((currentDATE>=32) && (currentDATE<60)){
    PRIS=PRS1[2];
  }
  else if ((currentDATE>=60) && (currentDATE<91)){
    PRIS=PRS1[3];
  }
  else if ((currentDATE>=91) && (currentDATE<121)){
    PRIS=PRS1[4];
  }
  else if ((currentDATE>=121) && (currentDATE<152)){
  }
}
PRIS=PRS1[5];
}
else if ((currentDATE>=152) && (currentDATE<182)){
PRIS=PRS1[6];
}
else if ((currentDATE>=182) && (currentDATE<213)){
PRIS=PRS1[7];
}
else if ((currentDATE>=213) && (currentDATE<244)){
PRIS=PRS1[8];
}
else if ((currentDATE>=244) && (currentDATE<274)){
PRIS=PRS1[9];
}
else if ((currentDATE>=274) && (currentDATE<305)){
PRIS=PRS1[10];
}
else if ((currentDATE>=305) && (currentDATE<335)){
PRIS=PRS1[11];
}
else if ((currentDATE>=335) && (currentDATE<=365)){
PRIS=PRS1[12];
}
}
else if (YEAR==2){
if ((currentDATE>=1) && (currentDATE<32)){
PRIS=PRS2[1];
}
else if ((currentDATE>=32) && (currentDATE<60)){
PRIS=PRS2[2];
}
else if ((currentDATE>=60) && (currentDATE<91)){
PRIS=PRS2[3];
}
else if ((currentDATE>=91) && (currentDATE<121)){
PRIS=PRS2[4];
}
else if ((currentDATE>=121) && (currentDATE<152)){
PRIS=PRS2[5];
}
else if ((currentDATE>=152) && (currentDATE<182)){
PRIS=PRS2[6];
}
else if ((currentDATE>=182) && (currentDATE<213)){
PRIS=PRS2[7];
}
else if ((currentDATE>=213) && (currentDATE<244)){
PRIS=PRS2[8];
}
else if ((currentDATE>=244) && (currentDATE<274)){
PRIS=PRS2[9];
}
else if ((currentDATE>=274) && (currentDATE<305)){
    PRIS=PRS2[10];
}
else if ((currentDATE>=305) && (currentDATE<335)){
    PRIS=PRS2[11];
}
else if ((currentDATE>=335) && (currentDATE<=365)){
    PRIS=PRS2[12];
}
else if (YEAR==3){
    if ((currentDATE>=1) && (currentDATE<32)){
        PRIS=PRS3[1];
    }
    else if ((currentDATE>=32) && (currentDATE<60)){
        PRIS=PRS3[2];
    }
    else if ((currentDATE>=60) && (currentDATE<91)){
        PRIS=PRS3[3];
    }
    else if ((currentDATE>=91) && (currentDATE<121)){
        PRIS=PRS3[4];
    }
    else if ((currentDATE>=121) && (currentDATE<152)){
        PRIS=PRS3[5];
    }
    else if ((currentDATE>=152) && (currentDATE<182)){
        PRIS=PRS3[6];
    }
    else if ((currentDATE>=182) && (currentDATE<213)){
        PRIS=PRS3[7];
    }
    else if ((currentDATE>=213) && (currentDATE<244)){
        PRIS=PRS3[8];
    }
    else if ((currentDATE>=244) && (currentDATE<274)){
        PRIS=PRS3[9];
    }
    else if ((currentDATE>=274) && (currentDATE<305)){
        PRIS=PRS3[10];
    }
    else if ((currentDATE>=305) && (currentDATE<335)){
        PRIS=PRS3[11];
    }
    else if ((currentDATE>=335) && (currentDATE<=365)){
        PRIS=PRS3[12];
    }
}
if ((animal->SEX==0) && (animal->REPLA==0) && (animal->AGE >= animal->WEAN) && (animal->WEAN>0 )){
if (animal->CDWT < 400) {
    HFRV=(PRIS*PRH[4])/100 * animal->CDWT;
    THFR+=HFRV;
    if (YEAR ==0) {
        THFR1+=HFRV;
        NHFC1++;  
    } 
    NHFC++;
    TWHS+=animal->CDWT;
    NumAvailRepla--;
} else if ((animal->CDWT >= 400) && (animal->CDWT < 500)){
    HFRV=(PRIS*PRH[5])/100 * animal->CDWT;
    THFR+=HFRV;
    if (YEAR ==0) {
        THFR1+=HFRV;
        NHFC1++;  
    } 
    NHFC++;
    TWHS+=animal->CDWT;
    NumAvailRepla--;
} else if ((animal->CDWT >= 500) && (animal->CDWT < 600)){
    HFRV=(PRIS*PRH[6])/100 * animal->CDWT;
    THFR+=HFRV;
    if (YEAR ==0) {
        THFR1+=HFRV;
        NHFC1++;  
    } 
    NHFC++;
    TWHS+=animal->CDWT;
    NumAvailRepla--;
} else if ((animal->CDWT >= 600) && (animal->CDWT < 700)){
    HFRV=(PRIS*PRH[7])/100 * animal->CDWT;
    THFR+=HFRV;
    if (YEAR ==0) {
        THFR1+=HFRV;
        NHFC1++;  
    } 
    NHFC++;
    TWHS+=animal->CDWT;
    NumAvailRepla--;
} else if ((animal->CDWT >= 700) && (animal->CDWT < 800)){
    HFRV=(PRIS*PRH[8])/100 * animal->CDWT;
    THFR+=HFRV;
    if (YEAR ==0) {
        THFR1+=HFRV;
        NHFC1++;  
    } 
}
NHFC++;
TWHS+=animal->CDWT;
NumAvailRepla--;
}
else if ((animal->CDWT >= 800) && (animal->CDWT < 900)) {
    HFRV=(PRIS*PRH[9])/100 * animal->CDWT;
    THFR+=HFRV;
    if (YEAR ==0) {
        THFR1+=HFRV;
        NHFC1++;
    }
    NHFC++;
    TWHS+=animal->CDWT;
    NumAvailRepla--;
}
else if ((animal->CDWT >= 900)) {
    HFRV=(PRIS*PRH[10])/100 * animal->CDWT;
    THFR+=HFRV;
    if (YEAR ==0) {
        THFR1+=HFRV;
        NHFC1++;
    }
    NHFC++;
    TWHS+=animal->CDWT;
    NumAvailRepla--;
}
if (animal->BREED == 1) {
    BDH=HFRV * PREH;
    TBDH+=BDH;
    if (YEAR ==0) {
        TBDH1+=BDH;
    }
}
else if ((animal->BREED > 1) && (animal->COLR==0) && (animal->PURE != 0)) {
    BDH=HFRV * PREHC;
    TBDH+=BDH;
    if (YEAR ==0) {
        TBDH1+=BDH;
    }
}
else if ((animal->BREED > 5) && (animal->COLR==0) && (animal->PURE == 0)) {
    BDH=HFRV * PREHC;
    TBDH+=BDH;
    if (YEAR ==0) {
        TBDH1+=BDH;
    }
}
if (SALH==0) {
    SHFC=HFRV*DPREM;
TDSPH+=SHFC;
if (YEAR ==0){
    TDSPH1+=SHFC;
}
SHKH=SHKHP;
TSHKH+=SHKH;
ASHKH=TSHKH/NHFC;
if (YEAR ==0){
    ASHKH1=TSHKH/NHFC;
}
else if ((SALH >=1) && (DIST <= 50) && (ONIT==0)){
    SHKH=.022;
    TSHKH+=SHKH;
    ASHKH=TSHKH/NHFC;
    if (YEAR ==0){
        ASHKH1=TSHKH/NHFC;
    }
    FECA=FEE;
    YRD=HFRV*YRCD;
    YDCA+=YRD;
    TFECA+=FECA;
    if (YEAR ==0){
        YDCA1+=YRD;
        TFECA1+=FECA;
    }
}
else if ((SALH >=1) && (DIST <= 50) && (ONIT==1)){
    SHKH=.042;
    TSHKH+=SHKH;
    ASHKH=TSHKH/NHFC;
    if (YEAR ==0){
        ASHKH1=TSHKH/NHFC;
    }
    FECA=FEE;
    YRD=HFRV*YRCD;
    YDCA+=YRD;
    TFECA+=FECA; if (YEAR ==0){
        YDCA1+=YRD;
        TFECA1+=FECA;
    }
}
else if ((SALH >=1) && (DIST > 50) && (ONIT==0)){
    SHKH=.022 + (.00308 * (DIST/50));
    TSHKH+=SHKH;
    ASHKH=TSHKH/NHFC;
    if (YEAR ==0){
        ASHKH1=TSHKH/NHFC;
    }
    FECA=FEE;
    YRD=HFRV*YRCD;
YDCA+=YRD;
TFECA+=FECA;if (YEAR ==0){
YDCA1+=YRD;
TFECA1+=FECA;
}
else if ((SALH >=1) && (DIST > 50) && (ONIT==1)){
SHKH=.042 + (.00308 * (DIST/50));
TSHKH+=SHKH;
ASHKH=TSHKH/NHFC;
if (YEAR ==0){
ASHKH1=TSHKH/NHFC;
}
FECA=FEE;
YRD=HFRV*YRCD;
YDCA+=YRD;
TFECA+=FECA;
if (YEAR ==0){
YDCA1+=YRD;
TFECA1+=FECA;
}
if (((SALH > 1) && (animal->CDWT < 600))
)
FECA=TELO+VACA+GFEE+LASS-AGFEE;
TFECA+=FECA;
if (YEAR ==0){
YDCA1+=YRD;
TFECA1+=FECA;
}
else if ((SALH > 1) && (animal->CDWT >= 600)){
FECA=TELO+VACA+GFEE+LASS+FEEA-AGFEE;
TFECA+=FECA;
if (YEAR ==0){
YDCA1+=YRD;
TFECA1+=FECA;
}
if (SALH==2){
SHFE=HFRV*EPREM;
TESPH+=SHFE;
if (YEAR ==0){
TESPH1+=SHFE;
}
}
else if ((animal->SEX == 2) && (animal->AGE >= animal->WEAN)
&& (animal->WEAN>0)){
if (animal->CDWT < 400){
STRV=((PRIS*PRS[4])/100)*animal->CDWT;
TSTR+=STRV;
if (YEAR ==0){
    TSTR1+=STRV;
    NSTC1++;
}
NSTC++;
TWSS+=animal->CDWT;
}
else if ((animal->CDWT >= 400) && (animal->CDWT < 500)){
    STRV=((PRIS*PRS[5])/100)*animal->CDWT;
    TSTR+=STRV;
    if (YEAR ==0){
        TSTR1+=STRV;
        NSTC1++;
    }
    NSTC++;
    TWSS+=animal->CDWT;
}
else if ((animal->CDWT >= 500) && (animal->CDWT < 600)){
    STRV=((PRIS*PRS[6])/100)*animal->CDWT;
    TSTR+=STRV;
    if (YEAR ==0){
        TSTR1+=STRV;
        NSTC1++;
    }
    NSTC++;
    TWSS+=animal->CDWT;
}
else if ((animal->CDWT >= 600) && (animal->CDWT < 700)){
    STRV=((PRIS*PRS[7])/100)*animal->CDWT;
    TSTR+=STRV;
    if (YEAR ==0){
        TSTR1+=STRV;
        NSTC1++;
    }
    NSTC++;
    TWSS+=animal->CDWT;
}
else if ((animal->CDWT >= 700) && (animal->CDWT < 800)){
    STRV=((PRIS*PRS[8])/100)*animal->CDWT;
    TSTR+=STRV;
    if (YEAR ==0){
        TSTR1+=STRV;
        NSTC1++;
    }
    NSTC++;
    TWSS+=animal->CDWT;
}
else if ((animal->CDWT >= 800) && (animal->CDWT < 900)){
    STRV=((PRIS*PRS[9])/100)*animal->CDWT;
    TSTR+=STRV;
    if (YEAR ==0){

TSTR1+=STRV;
NSTC1++;
}
NSTC++;
TWSS+=animal->CDWT;
}
else if ((animal->CDWT >= 900)){
STRV=((PRIS*PRS[10])/100)*animal->CDWT;
TSTR+=STRV;
if (YEAR ==0){
TSTR1+=STRV;
NSTC1++;
}
NSTC++;
TWSS+=animal->CDWT;
}
if (animal->BREED == 1){
BDS=STRV * PRES;
TBDS+=BDS;
if (YEAR ==0){
TBDS1+=BDS;
}
}
else if ((animal->BREED > 1) && (animal->COLR==0) && (animal->PURE !=0) ){
BDS=STRV * PRES;
TBDS+=BDS;
if (YEAR ==0){
TBDS1+=BDS;
}
}
else if ((animal->BREED > 5) && (animal->COLR==0) && (animal->PURE ==0) ){
BDS=STRV * PRES;
TBDS+=BDS;
if (YEAR ==0){
TBDS1+=BDS;
}
}
if (SALS==0){
SSC=STRV*DPREM;
TDSPS+=SSC;
if (YEAR ==0){
TDSPS1+=SSC;
}
SHKS=SHKSP;
TSHKS+=SHKS;
ASHKS=TSHKS/NSTC;
if (YEAR ==0){
ASHKS1=TSHKS/NSTC;
}
```c
} else if ((SALS >=1) && (DIST <= 50) && (ONIT==0)) {
    SHKS=.022;
    TSHKS+=SHKS;
    ASHKS=TSHKS/NSTC;
    if (YEAR ==0) {
        ASHKS1=TSHKS/NSTC;
    }
    FECA=FEE;
    YRD=STRV*YRCD;
    YDCA+=YRD;
    TFECA+=FECA;
    if (YEAR ==0) {
        YDCA1+=YRD;
        TFECA1+=FECA;
    }
} else if ((SALH >=1) && (DIST <= 50) && (ONIT==1)) {
    SHKS=.042;
    TSHKS+=SHKS;
    ASHKS=TSHKS/NSTC;
    if (YEAR ==0) {
        ASHKS1=TSHKS/NSTC;
    }
    FECA=FEE;
    YRD=STRV*YRCD;
    YDCA+=YRD;
    TFECA+=FECA;
    if (YEAR ==0) {
        YDCA1+=YRD;
        TFECA1+=FECA;
    }
} else if ((SALH >=1) && (DIST > 50) && (ONIT==0)) {
    SHKS=.022 + (.00308 * (DIST/50));
    TSHKS+=SHKS;
    ASHKS=TSHKS/NSTC;
    if (YEAR ==0) {
        ASHKS1=TSHKS/NSTC;
    }
    FECA=FEE;
    YRD=STRV*YRCD;
    YDCA+=YRD;
    TFECA+=FECA;
    if (YEAR ==0) {
        YDCA1+=YRD;
        TFECA1+=FECA;
    }
} else if ((SALH >=1) && (DIST > 50) && (ONIT==1)) {
    SHKS=.042 + (.00308 * (DIST/50));
```
TSHKS+=SHKS;
ASHKS=TSHKS/NSTC;
if (YEAR ==0){
ASHKS1=TSHKS/NSTC;
}
FECA=FEE;
YRD=STRV*YRCD;
YDCA+=YRD;
TFECA+=FECA;
if (YEAR ==0){
YDCA1+=YRD;
TFECA1+=FECA;
}
if ((SALS > 1) && (animal->CDWT < 600)){
FECA=TELO+VACA+GTEE+LAGSS-AGFEE;
TFECA+=FECA;
if (YEAR ==0){
YDCA1+=YRD;
TFECA1+=FECA;
}
else if ((SALH > 1) && (animal->CDWT >= 600)){
FECA=TELO+VACA+GTEE+LAAE-AGFEE;
TFECA+=FECA;
if (YEAR ==0){
YDCA1+=YRD;
TFECA1+=FECA;
}
}
if (SALS==2){
SSE=STRV*PREM;
TESPS+=SSE;
if (YEAR ==0){
TESPS1+=SSE;
}
}
}

int Market::SellStockerCalves(AnimalRecord *animal){
double BDSH;
double BDSS;
double FECAS=0;
double FEE=4.50;
double FEEA=1.00;
double GTEE=.75;
double AGFEE=.50;
double LASS=.25;
double PREH=.00427;
double PREHC=.01485;
double PRES=.02153;
double PRESC=.01672;
double SHFRV=0;
double SHKSH=0;
double SHKSS=0;
double SHKSHP=.03;
double SHKSSP=.03;
double SSHFC=0;
double SSHFE=0;
double SSSC=0;
double SSSE=0;
double SSTRV=0;
double TELO=1.5;
double VACA=1.00;
double YRCD=.03;
double YRDS=0;
double DPREM=.05;
double EPREM=.065;

double SPRS1[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
double SPRS2[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
double SPRS3[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};

double SPA[13]={0,1.015,1.045,1.054,1.041,1.019,1.011,0.997,1.009,0.968,0.948,0.942,0.950};

double SPRH[11]={0,0,0,0,0.9237,0.9344,0.9316,0.9077,0.8896,0.8594,0.8229};

double SPRS[13]={0,0,0,0,1.139,1.095,1.082,1.029,1.100,0.967,0.926};

SPRS1[1]=(SPRSA1*SPA[1]);
SPRS1[2]=(SPRSA1*SPA[2]);
SPRS1[3]=(SPRSA1*SPA[3]);
SPRS1[4]=(SPRSA1*SPA[4]);
SPRS1[5]=(SPRSA1*SPA[5]);
SPRS1[6]=(SPRSA1*SPA[6]);
SPRS1[7]=(SPRSA1*SPA[7]);
SPRS1[8]=(SPRSA1*SPA[8]);
SPRS1[9]=(SPRSA1*SPA[9]);
SPRS1[10]=(SPRSA1*SPA[10]);
SPRS1[12]=(SPRSA1*SPA[12]);
SPRS2[1]=(SPRSA2*SPA[1]);
SPRS2[2]=(SPRSA2*SPA[2]);
SPRS2[3]=(SPRSA2*SPA[3]);
SPRS2[4]=(SPRSA2*SPA[4]);
SPRS2[5]=(SPRSA2*SPA[5]);
SPRS2[6]=(SPRSA2*SPA[6]);
SPRS2[7]=(SPRSA2*SPA[7]);
SPRS2[8]=(SPRSA2*SPA[8]);
SPRS2[9]=(SPRSA2*SPA[9]);
SPRS2[10]=(SPRSA2*SPA[10]);
SPRS2[12] = (SPRSA2 * SPA[12]);
SPRS3[1] = (SPRSA3 * SPA[1]);
SPRS3[2] = (SPRSA3 * SPA[2]);
SPRS3[3] = (SPRSA3 * SPA[3]);
SPRS3[4] = (SPRSA3 * SPA[4]);
SPRS3[5] = (SPRSA3 * SPA[5]);
SPRS3[6] = (SPRSA3 * SPA[6]);
SPRS3[7] = (SPRSA3 * SPA[7]);
SPRS3[8] = (SPRSA3 * SPA[8]);
SPRS3[9] = (SPRSA3 * SPA[9]);
SPRS3[10] = (SPRSA3 * SPA[10]);
SPRS3[12] = (SPRSA3 * SPA[12]);

if (YEAR==1){
    if ((currentDATE>=1) && (currentDATE<32)){
        PRIS = SPRS1[1];
    }
    else if ((currentDATE>=32) && (currentDATE<60)){
        PRIS = SPRS1[2];
    }
    else if ((currentDATE>=60) && (currentDATE<91)){
        PRIS = SPRS1[3];
    }
    else if ((currentDATE>=91) && (currentDATE<121)){
        PRIS = SPRS1[4];
    }
    else if ((currentDATE>=121) && (currentDATE<152)){
        PRIS = SPRS1[5];
    }
    else if ((currentDATE>=152) && (currentDATE<182)){
        PRIS = SPRS1[6];
    }
    else if ((currentDATE>=182) && (currentDATE<213)){
        PRIS = SPRS1[7];
    }
    else if ((currentDATE>=213) && (currentDATE<244)){
        PRIS = SPRS1[8];
    }
    else if ((currentDATE>=244) && (currentDATE<274)){
        PRIS = SPRS1[9];
    }
    else if ((currentDATE>=274) && (currentDATE<305)){
        PRIS = SPRS1[10];
    }
    else if ((currentDATE>=305) && (currentDATE<335)){
        PRIS = SPRS1[11];
    }
    else if ((currentDATE>=335) && (currentDATE<=365)){
        PRIS = SPRS1[12];
    }
}
else if (YEAR==2) {
    if ((currentDATE>=1) && (currentDATE<32)){
        PRIS=SPRS2[1];
    }
    else if ((currentDATE>=32) && (currentDATE<60)){
        PRIS=SPRS2[2];
    }
    else if ((currentDATE>=60) && (currentDATE<91)){
        PRIS=SPRS2[3];
    }
    else if ((currentDATE>=91) && (currentDATE<121)){
        PRIS=SPRS2[4];
    }
    else if ((currentDATE>=121) && (currentDATE<152)){
        PRIS=SPRS2[5];
    }
    else if ((currentDATE>=152) && (currentDATE<182)){
        PRIS=SPRS2[6];
    }
    else if ((currentDATE>=182) && (currentDATE<213)){
        PRIS=SPRS2[7];
    }
    else if ((currentDATE>=213) && (currentDATE<244)){
        PRIS=SPRS2[8];
    }
    else if ((currentDATE>=244) && (currentDATE<274)){
        PRIS=SPRS2[9];
    }
    else if ((currentDATE>=274) && (currentDATE<305)){
        PRIS=SPRS2[10];
    }
    else if ((currentDATE>=305) && (currentDATE<335)){
        PRIS=SPRS2[11];
    }
    else if ((currentDATE>=335) && (currentDATE<=365)){
        PRIS=SPRS2[12];
    }
}
else if (YEAR==3) {
    if ((currentDATE>=1) && (currentDATE<32)){
        PRIS=SPRS3[1];
    }
    else if ((currentDATE>=32) && (currentDATE<60)){
        PRIS=SPRS3[2];
    }
    else if ((currentDATE>=60) && (currentDATE<91)){
        PRIS=SPRS3[3];
    }
    else if ((currentDATE>=91) && (currentDATE<121)){
        PRIS=SPRS3[4];
    }
else if ((currentDATE>=121) && (currentDATE<152)){
    PRIS=SPRS3[5];
}
else if ((currentDATE>=152) && (currentDATE<182)){
    PRIS=SPRS3[6];
}
else if ((currentDATE>=182) && (currentDATE<213)){
    PRIS=SPRS3[7];
}
else if ((currentDATE>=213) && (currentDATE<244)){
    PRIS=SPRS3[8];
}
else if ((currentDATE>=244) && (currentDATE<274)){
    PRIS=SPRS3[9];
}
else if ((currentDATE>=274) && (currentDATE<305)){
    PRIS=SPRS3[10];
}
else if ((currentDATE>=305) && (currentDATE<335)){
    PRIS=SPRS3[11];
}
else if ((currentDATE>=335) && (currentDATE<=365)){
    PRIS=SPRS3[12];
}

if ( (animal->WEAN>0) && (animal->SEX>3) && (animal->REPLA==0) ){
    SELL=((animal->WEAN) + DRET);
}
if ((animal->SEX==4) && (animal->REPLA==0) && (animal->AGE>=SELL)){
    if ((animal->CDWT <= 400) ){
        SHFRV=(PRIS*SPRH[4])/100 * animal->CDWT;
        TSHFR+=SHFRV;
        NSHFC++;
        TWSHS+=animal->CDWT;
        TotWeanWtH+=animal->WEANWT;
        TNSCOH--;
        if ((animal->C AlfSEAS==1) && (animal->SEX>3)){
            SSELLDATE+=currentDATE;
            SSCALF++;
        }
    }
    else if ((animal->C AlfSEAS==2) && (animal->SEX>3)){
        FSELLDATE+=currentDATE;
        FSCALF++;
    }
}
else if ((animal->CDWT >= 400) && (animal->CDWT < 500)){
    SHFRV=(PRIS*SPRH[5])/100 * animal->CDWT;
    TSHFR+=SHFRV;
NSHFC++;  
TWSHS+=animal->CDWT;  
TotWeanWtH+=animal->WEANWT;  
TNSCOH--;  
if ( ((animal->CALFSEAS==1) && (animal->SEX>3))){  
  SSELLDATE+=currentDATE;  
  SSCALLF++;  
}  
else if ( ((animal->CALFSEAS==2) && (animal->SEX>3))){  
  FSELLDATE+=currentDATE;  
  FSCALLF++;  
}  
else if ( ((animal->CDWT >= 500) && (animal->CDWT < 600))){  
  SHFRV=(PRIS*SPRH[6])/100 * animal->CDWT;  
  TSHFR+=SHFRV;  
  NSHFC++;  
  TWSHS+=animal->CDWT;  
  TotWeanWtH+=animal->WEANWT;  
  TNSCOH--;  
  if ( ((animal->CALFSEAS==1) && (animal->SEX>3))){  
    SSELLDATE+=currentDATE;  
    SSCALLF++;  
  }  
  else if ( ((animal->CALFSEAS==2) && (animal->SEX>3))){  
    FSELLDATE+=currentDATE;  
    FSCALLF++;  
  }  
}  
else if ( ((animal->CDWT >= 600) && (animal->CDWT < 700))){  
  SHFRV=(PRIS*SPRH[7])/100 * animal->CDWT;  
  TSHFR+=SHFRV;  
  NSHFC++;  
  TWSHS+=animal->CDWT;  
  TotWeanWtH+=animal->WEANWT;  
  TNSCOH--;  
  if ( ((animal->CALFSEAS==1) && (animal->SEX>3))){  
    SSELLDATE+=currentDATE;  
    SSCALLF++;  
  }  
  else if ( ((animal->CALFSEAS==2) && (animal->SEX>3))){  
    FSELLDATE+=currentDATE;  
    FSCALLF++;  
  }  
}  
else if ( ((animal->CDWT >= 700) && (animal->CDWT < 800))){  
  SHFRV=(PRIS*SPRH[8])/100 * animal->CDWT;  
  TSHFR+=SHFRV;  
  NSHFC++;  
  TWSHS+=animal->CDWT;  
  TotWeanWtH+=animal->WEANWT;  
}
TNSCOH--; 
if ((animal->C alfSEAS==1) && (animal->SEX>3)) {
    SSELLDATE+=currentDATE;
    SSCALF++;
} else if ((animal->C alfSEAS==2) && (animal->SEX>3)) {
    FSELLDATE+=currentDATE;
    FSCALF++;
} 
} else if ((animal->CDWT >= 800) && (animal->CDWT < 900)) {
    SHFRV=(PRIS*SPRH[9])/100 * animal->CDWT;
    TSHFR+=SHFRV;
    NSHFC++;
    TWSHS+=animal->CDWT;
    TotWeanWtH+=animal->WEANWT;
    TNSCOH--; 
    if ((animal->C alfSEAS==1) && (animal->SEX>3)) {
        SSELLDATE+=currentDATE;
        SSCALF++;
    } else if ((animal->C alfSEAS==2) && (animal->SEX>3)) {
        FSELLDATE+=currentDATE;
        FSCALF++;
    } 
} else if ((animal->CDWT >= 900) ) {
    SHFRV=(PRIS*SPRH[10])/100 * animal->CDWT;
    TSHFR+=SHFRV;
    NSHFC++;
    TWSHS+=animal->CDWT;
    TotWeanWtH+=animal->WEANWT;
    TNSCOH--; 
    if ((animal->C alfSEAS==1) && (animal->SEX>3)) {
        SSELLDATE+=currentDATE;
        SSCALF++;
    } else if ((animal->C alfSEAS==2) && (animal->SEX>3)) {
        FSELLDATE+=currentDATE;
        FSCALF++;
    } 
} 
if (animal->BREED == 1){
    BDSH=SHFRV * PREH;
    TBDSH+=BDSH;
} } else if ((animal->BREED > 1) && (animal->COLR==0) && (animal->PURE !=0)){
    BDSH=SHFRV * PREH;
    TBDSH+=BDSH;
}
else if ((animal->BREED > 5) && (animal->COLR==0) && (animal->PURE ==0)){
    BDSH=SHFRV * PREH;
    TBDSH+=BDSH;
}
if (SALSH==0){
    SSHFC=SHFRV*DPREM;
    TDSPSH+=SSHFC;
    SHKSH=SHKSHP;
    TSHKSH+=SHKSH;
    ASHKSH=TSHKSH/NSHFC;
}
else if ((SALH >=1) && (DIST <= 50) && (ONIT==0)){
    SHKSH=.022;
    TSHKSH+=SHKSH;
    ASHKSH=TSHKSH/NSHFC;
    FECAS=FEE;
    YRDS=SHFRV*YRCD;
    YDCAS+=YRDS;
    TFECAS+=FECAS;
}
else if ((SALH >=1) && (DIST <= 50) && (ONIT==1)){
    SHKSH=.042;
    TSHKSH+=SHKSH;
    ASHKSH=TSHKSH/NSHFC;
    FECAS=FEE;
    YRDS=SHFRV*YRCD;
    YDCAS+=YRDS;
    TFECAS+=FECAS;
}
else if ((SALH >=1) && (DIST > 50) && (ONIT==0)){
    SHKSH=.022 + (.00308 * (DIST/50));
    TSHKSH+=SHKSH;
    ASHKSH=TSHKSH/NSHFC;
    FECAS=FEE;
    YRDS=SHFRV*YRCD;
    YDCAS+=YRDS;
    TFECAS+=FECAS;
}
else if ((SALH >=1) && (DIST > 50) && (ONIT==1)){
    SHKSH=.042 + (.00308 * (DIST/50));
    TSHKSH+=SHKSH;
    ASHKSH=TSHKSH/NSHFC;
    FECAS=FEE;
    YRDS=SHFRV*YRCD;
    YDCAS+=YRDS;
    TFECAS+=FECAS;
}
if ((SALSH > 1) && (animal->CDWT >= 600)){
    FECAS=TELO+VACA+GFEE+LASS+FEEA-AGFEE;
    TFECAS+=FECAS;
}
else if ((SALSH > 1) && (animal->CDWT < 600)){
    FECAS=TELO+VACA+GFEE+LASS-AGFEE;
    TFECAS+=FECAS;
}
if (SALSH==2){
    SSHFE=SHFRV*EPREM;
    TESPSh+=SSHFE;
}
else if ((animal->SEX == 5) && (animal->AGE >= SELL)){
    if ((animal->CDWT <= 400)){
        SSTRV=((PRIS*SPRS[4])/100)*animal->CDWT;
        TSSTR+=SSTRV;
        NSSTC++;
        TWSSS=animal->CDWT;
        TotWeanWtS+=animal->WEANWT;
        TNSCOH--;
        if ((animal->CALFSEAS==1) && (animal->SEX>3)){
            SSELLDATE+=currentDATE;
            SSCALF++;
        }
        else if ((animal->CALFSEAS==2) && (animal->SEX>3)){
            FSELLDATE+=currentDATE;
            FSCALF++;
        }
    }
    else if ((animal->CDWT >= 400) && (animal->CDWT < 500)){
        SSTRV=((PRIS*SPRS[5])/100)*animal->CDWT;
        TSSTR+=SSTRV;
        NSSTC++;
        TWSSS=animal->CDWT;
        TotWeanWtS+=animal->WEANWT;
        TNSCOH--;
        if ((animal->CALFSEAS==1) && (animal->SEX>3)){
            SSELLDATE+=currentDATE;
            SSCALF++;
        }
        else if ((animal->CALFSEAS==2) && (animal->SEX>3)){
            FSELLDATE+=currentDATE;
            FSCALF++;
        }
    }
    else if ((animal->CDWT >= 500) && (animal->CDWT < 600)){
        SSTRV=((PRIS*SPRS[6])/100)*animal->CDWT;
        TSSTR+=SSTRV;
        NSSTC++;
        TWSSS=animal->CDWT;
        TotWeanWtS+=animal->WEANWT;
        TNSCOH--;
        if ((animal->CALFSEAS==1) && (animal->SEX>3)){
            SSELLDATE+=currentDATE;
            SSCALF++;
        }
    }
}
SELLDATE+=currentDATE;
SSCALF++;
}
else if ((animal->CALFSEAS==2) && (animal->SEX>3)){
    FSELLDATE+=currentDATE;
    FSCALF++;
}
}
else if ((animal->CDWT >= 600) && (animal->CDWT < 700)){
    SSTRV=((PRIS*SPRS[7])/100)*animal->CDWT;
    TSSTR+=SSTRV;
    NSSTC++;
    TWSSS+=animal->CDWT;
    TotWeanWtS+=animal->WEANWT;
    TNSCOH--;
    if ((animal->CALFSEAS==1) && (animal->SEX>3)){
        SSELLDATE+=currentDATE;
        SSCALF++;
    }
}
else if ((animal->CALFSEAS==2) && (animal->SEX>3)){
    FSELLDATE+=currentDATE;
    FSCALF++;
}
}
else if ((animal->CDWT >= 700) && (animal->CDWT < 800)){
    SSTRV=((PRIS*SPRS[8])/100)*animal->CDWT;
    TSSTR+=SSTRV;
    NSSTC++;
    TWSSS+=animal->CDWT;
    TotWeanWtS+=animal->WEANWT;
    TNSCOH--;
    if ((animal->CALFSEAS==1) && (animal->SEX>3)){
        SSELLDATE+=currentDATE;
        SSCALF++;
    }
}
else if ((animal->CALFSEAS==2) && (animal->SEX>3)){
    FSELLDATE+=currentDATE;
    FSCALF++;
}
}
else if ((animal->CDWT >= 800) && (animal->CDWT < 900)){
    SSTRV=((PRIS*SPRS[9])/100)*animal->CDWT;
    TSSTR+=SSTRV;
    NSSTC++;
    TWSSS+=animal->CDWT;
    TotWeanWtS+=animal->WEANWT;
    TNSCOH--;
    if ((animal->CALFSEAS==1) && (animal->SEX>3)){
        SSELLDATE+=currentDATE;
        SSCALF++;
    }
}
else if ((animal->CALFSEAS==2) && (animal->SEX>3)){
  FSELLDATE+=currentDATE;
  FSCALF++;
}
else if ((animal->CDWT >= 900)){
  SSTRV=((PRIS*SPRS[10])/100)*animal->CDWT;
  TSSTR+=SSTRV;
  NSSTC++;
  TWSSS+=animal->CDWT;
  TotWeanWtS+=animal->WEANWT;
  TNSCOH--;
  if ((animal->CALFSEAS==1) && (animal->SEX>3)){
    SSELLDATE+=currentDATE;
    SSCALF++;
  }
  else if ((animal->CALFSEAS==2) && (animal->SEX>3)){
    FSELLDATE+=currentDATE;
    FSCALF++;
  }
  if (animal->BREED == 1){
    BDSS=SSTRV * PRES;
    TBDSS+=BDSS;
  }
  else if ((animal->BREED > 1) && (animal->COLR==0) && (animal->PURE != 0)){
    BDSS=SSTRV * PRESC;
    TBDSS+=BDSS;
  }
  else if ((animal->BREED > 5) && (animal->COLR==0) && (animal->PURE==0)){
    BDSS=SSTRV * PRESC;
    TBDSS+=BDSS;
  }
  if (SALSS==0){
    SSSC=SSTRV*DPREM;
    TDSPSS+=SSSC;
    SHKSS=SHKSSP;
    TSHKSS+=SHKSS;
    ASHKSS=TSHKSS/NSSTC;
  }
  else if ((SALSS >=1) && (DIST <= 50) && (ONIT==0)){
    SHKSS=.022;
    TSHKSS+=SHKSS;
    ASHKSS=TSHKSS/NSSTC;
    FECAS=FEE;
    YRDS=SSTRV*YRCD;
    YDCAS+=YRDS;
    TFECAS+=FECAS;
}
else if ((SALSS >=1) && (DIST <= 50) && (ONIT==1)){
    SHKSS=.042;
    TSHKSS+=SHKSS;
    ASHKSS=TSHKSS/NSSTC;
    FECAS=FEE;
    YRDS=SSTRV*YRCD;
    YDCAS+=YRDS;
    TFECAS+=FECAS;
}
else if ((SALSS >=1) && (DIST > 50) && (ONIT==0)){
    SHKSS=.022 + (.00308 * (DIST / 50));
    TSHKSS+=SHKSS;
    ASHKSS=TSHKSS/NSSTC;
    FECAS=FEE;
    YRDS=SSTRV*YRCD;
    YDCAS+=YRDS;
    TFECAS+=FECAS;
}
else if ((SALSS >=1) && (DIST > 50) && (ONIT==1)){
    SHKSS=.042 + (.00308 * (DIST/50));
    TSHKSS+=SHKSS;
    ASHKSS=TSHKSS/NSSTC;
    FECAS=FEE;
    YRDS=SSTRV*YRCD;
    YDCAS+=YRDS;
    TFECAS+=FECAS;
}
if ((SALSS > 1) && (animal->CDWT >= 600)){
    FECAS+=TELO+VACA+GFEE+LASS+FEEA;
    TFECAS+=FECAS;
}
if (SALSS==2){
    SSSE=SSTRV*EPREM;
    TESPSS+=SSSE;
}
if ( (animal->AGE >= SELL) && ( ( (animal->SEX==4) && (animal->REPLA==0) ) || (animal->SEX==5) ) ){ return (animal->ID); }
return -1;

void Market::BudgetReport(ofstream& outfile){
double AWCC=0;
double AWSS=0;
double AWHS=0;
double AWSSS=0;
double AWSHS=0;
double ADGSS=0;
double ADGSH=0;
double DRATS=0;
double ADGSCS=0;
double AVERAGE=0;
int CNTR=50000;
double IPCS=1.8;
double SALT=68;
double TOEXS=0;
double TWCS = TWHS + TWSS;
double TNCS = NHFC + NSTC;
double CostSKCC = 0-(TCCR * ASHKC);
double CostSKHC = 0-(THFR * ASHKH);
double CostSKSC = 0-(TSTR * ASHKS);
double CostSKOC = 0 - (TOCR * ASHKOC);

TTSTR2=TTSTR1;
TTHFR2=TTHFR1;

if (YEAR == 1){
double CostSKHC0 = 0-(THFR1 * ASHKH1);
double CostSKSC0 = 0-(TSTR1 * ASHKS1);
TTSTR0 = TSTR1 + TBDS1 + TDSPS1 + TESPS1 + CostSKSC0;
TTHFR0 = THFR1 + TBDH1 + TDSPH1 + TESPH1 + CostSKHC0;
if (NSTC1>0){
ATTSTR0=TTSTR0/NSTC1;
}
else{
ATTSTR0=0;
}
if (NHFC1>0){
ATTHFR0=TTHFR0/NHFC1;
}
else{
ATTHFR0=0;
}
}
NSTC1=NSTC;
NHFC1=NHFC;
TWSS1=TWSS;
TTSTR1=TTSTR;
TTHFR1=TTHFR;

double TTSTR = TSTR + TBDS + TDSPS + TESPS + CostSKSC;
double TTHFR = THFR + TBDH + TDSPH + TESPH + CostSKHC;

/* More code here */
double LSUP = LSCS * COWS;
double BDEP = NBULLS * (BCST * (1-BSAL))/BUSE;
double TAIC = AICS * AvServConcp * (CNRAT/100) * COWS * ARTI;
double TPCA = TCSC * (NHFC + NSTC);
double TPCC = TCCS * NCCS;
double BFRP = BFRC * COWS;
double UTIL = UTLC * COWS;
double LABO = WAGE * HOUR * COWS;
double PRAX = LABO * FICA;
double MECO = MCCO * COWS;
double TOEX = CostofReplacements + TPRD + TPMN + FeedCost + SMFC + HDCE + IMFC + VMDC + LSUP + BDEP + TAIC + TPCA + TPCC + TFECA + YDCA + TFECAC + YDCAC + BFRP + UTIL + LABO + PRAX + MECO;
double INTE = (INTR/100) * ((TOEX-FeedCost) * LORT); if (INTE <0){
   INTE=0;
}
double TECCE = TOEX + INTE;
double NETIC = TPIC - TECCE;
if (NCCS>0){
   AWCC = TWCC/NCCS;
}
else{
   AWCC=0;
}
if (NSTC>0){
   AWSS=(TWSS/((double) NSTC));
}
else{
   AWSS=0;
}
if (NHFC>0){
   AWHS=(TWHS/((double) NHFC));
}
else{
   AWHS=0;
}
if (CNUMB>0){
   DEADC = (DNUM/((double) CNUMB));
}
else{
   DEADC=0;
}
double DRAT = (DNUMF/((double) COWS));
AVERAGE=(TOTALAGE/((double)(NumCowsYend-AgeBreeders[0]))) / 365;
/*STOCKER PARTIAL BUDGET DATA*/
double TWSCS=TWSHS + TWSSS;
double TNSCS=NSHFC + NSSTC;
double CostSKSHC = 0.0 - (TSHFR * ASHKSH);
double CostSKSSC = 0.0 - (TSSTR * ASHKSS);
double TTSSSTR = TSSTR + TBDSS + TDSPPS + TESPSS + CostSKSSC;
double TTSHFR = TSHFR + TBDSS + TDSPPS + TESPSS + CostSKSHC;
double HBIS = BASS * (TSWCS / (double) 100);
double TAPIS = TTSSSTR + TTSHFR + HBIS;
double TPRDS = CurrentSIZES * PRAS;
double TPMNS = CurrentSIZES * PMNS * (RESPS / (double) 100);
double SMFS = DMandSC * SCSS * DRET * TNSCS;
double HDCS = ((HDEX * TWSCS) / (double) CNTR) * HEDG2;
double IMPSC = (IPCS * NIPSS * (NSSTC + DNUMSS)) + (IPCS * 
NIPSH * (NSHFC + DNUMHS));
double INSSUS = INSURS * TNSCS;
double VMDCS = VMCSS * TNSCS;
double LSUPS = LSCSS * TNSCS;
double TPCSC = TCCSS * TNSCS;
double BFRPS = BFRS * TNSCS;
double UTILS = UTLCH * TNSCS;
double LABOS = WAGE * HOURS * TNSCS;
double PRAXS = LABOS * FICA;
double MESO = MCCOS * TNSCS;
if (HEDG2 == 0) {
    TAPIS = TTSSSTR + TTSHFR;
} else if (HEDG2 == 1) {
    TAPIS = TTSSSTR + TTSHFR + HBIS;
} else if (NumBreedSeasns == 1) {
    if ((SpringCalfSeasn == 1) && (DRET + WEANSDATE >= 365)) {
        if (YEAR == 1) {
            TOEXS = (ATTSTR0 * (NSSTC + DNUMSS)) + (ATTHFR0 * (NSHFC + 
DNUMHS)) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + 
IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + 
UTILS + LABOS + 
PRAXS + MESO;
        } else if (YEAR > 1) {
            TOEXS = TTSTR2 + TTHFR2 + TPRDS + TPMNS + STFeedCost + SMFS + 
HDCS + 
IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + 
UTILS + LABOS + 
PRAXS + MESO;
        }
    } else if ((SpringCalfSeasn == 1) && (DRET + WEANSDATE <= 365)) {
        TOEXS = TTSTR + TTHFR + TPRDS + TPMNS + STFeedCost + SMFS + 
HDCS + 
IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + 
UTILS + LABOS + 
PRAXS + MESO;
    }
}
else if ((FallCalfSeasn==1) && (DRET + WEANFDATE > 365)) {
    if (YEAR==1) {
        TOEXS = (ATTSTR0 * (NSSTC + DNUMSS)) + (ATTHFR0 * (NSHFC + DNUMHS)) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    } else if (YEAR>1) {
        TOEXS = TTSTR2 + TTHFR2 + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    }
} else if ((FallCalfSeasn==1) && (DRET + WEANFDATE <= 365)) {
    TOEXS = TTSTR + TTHFR + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
}
else if (NumBreedSeasns==2) {
    if (((SpringCalfSeasn==1) && (DRET + WEANSDATE>365)) && ((FallCalfSeasn==1) && (DRET + WEANFDATE>365))) {
        if (YEAR==1) {
            TOEXS = (ATTSTR0 * (NSSTC + DNUMSS)) + (ATTHFR0 * (NSHFC + DNUMHS)) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
        } else if (YEAR>1) {
            TOEXS = TTSTR2 + TTHFR2 + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
        }
    } else if (((SpringCalfSeasn==1) && (DRET + WEANSDATE<=365)) && ((FallCalfSeasn==1) && (DRET + WEANFDATE<=365))) {
        TOEXS = TTSTR + TTHFR + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    } else if (((SpringCalfSeasn==1) && (DRET + WEANSDATE>365)) && ((FallCalfSeasn==1) && (DRET + WEANFDATE<=365))) {
        TOEXS = TTSTR + TTHFR + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    } else if (((SpringCalfSeasn==1) && (DRET + WEANSDATE<=365)) && ((FallCalfSeasn==1) && (DRET + WEANFDATE>365))) {
        TOEXS = TTSTR + TTHFR + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    }
(FallCalfSeasn==1) && (DRET + WEANFDATE<=365))){
    if (YEAR==1){
        TOEXS = (((ATTSTR0 * (NSSTC + DNUMSS)) + (ATTHFR0 * (NSHFC + DNUMHS))) * (SSCALEF/(SSCALF+FSCALEF)))
            + (((TTSTR+TTHFR) * (FSCALEF/(SSCALF+FSCALEF)))) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    }
    else if (YEAR>1){
        TOEXS = (((TTSTR2 + TTHFR2) * (FSCALEF/(SSCALF+FSCALEF)))) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    }
}
else if (((SpringCalfSeasn==1) && (DRET + WEANFDATE<365)) && ((FallCalfSeasn==1) && (DRET + WEANFDATE>365))){
    if (YEAR==1){
        TOEXS = (((ATTSTR0 * (NSSTC + DNUMSS)) + (ATTHFR0 * (NSHFC + DNUMHS))) * (FSCALEF/(SSCALF+FSCALEF)))
            + (((TTSTR+TTHFR) * (SSCALEF/(SSCALF+FSCALEF)))) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    }
    else if (YEAR>1){
        TOEXS = (((TTSTR2 + TTHFR2) * (SSCALEF/(SSCALF+FSCALEF)))) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    }
}
else if (((SpringCalfSeasn==1) && (DRET + WEANSDATE<365)) && ((FallCalfSeasn==1) && (DRET + WEANFDATE>365))){
    if (YEAR==1){
        TOEXS = (((ATTSTR0 * (NSSTC + DNUMSS)) + (ATTHFR0 * (NSHFC + DNUMHS))) * (SSCALEF/(SSCALF+FSCALEF)))
            + (((TTSTR+TTHFR) * (FSCALEF/(SSCALF+FSCALEF)))) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    }
    else if (YEAR>1){
        TOEXS = (((TTSTR2 + TTHFR2) * (FSCALEF/(SSCALF+FSCALEF)))) + TPRDS + TPMNS + STFeedCost + SMFS + HDCS + IMPSC + VMDCS + LSUPS + TPCSC + TFECAS + YDCAS + BFRPS + UTILS + LABOS + PRAXS + MESO;
    }
}
}

double INTES=(((INTRS/100)/365)*DRET) * (TOEXS * LORTS);
double TESTE=TOEXS + INTES;
double NETICS=TAPIS-TESTE;
double TNCW=NSTC+NHFC;
if (TNCW>0){
double AWSS=TWSS/NSTC;
}
else{
    double AWSS=TWSS/NHFC;
}
double DNUMSC=DNUMHS + DNUMSS;
if (TNCW>0){

double DRATS=DNUMSC / (double) TNCW;
}
else{
    double DRATS=0;
}
if (DRET>0){
    ADGSH=((TWSHS - TotWeanWtH)/NSHFC) / ((double)DRET);
    ADGSS=((TWSSS - TotWeanWtS)/(double)NSSTC)/ ((double)DRET);
    ADGSCS=((((TWSHS + TWSSS) - (TotWeanWtH + TotWeanWtS)) / ((double) NSHFC + (double) NSSTC)) / ((double)DRET));
}
if (NSHFC>0) {
    AWSHS=TWSHS/(double)NSHFC;
} else{
    AWSHS=0;
}
if (NSSTC>0){
    AWSSS=TWSSS/(double) NSSTC;
} else{
    AWSSS=0;
}
if (NSSTC>0){
    ADGSS=(((TWSSS - TotWeanWtS)/NSSTC)/DRET);
} else{
    ADGSS = 0;
}
if (NSSTC>0){
    ADGSH=((TWSHS - TotWeanWtH)/NSHFC)/DRET);
} else{
    ADGSH=0;
}
if ((NSSTC>0) && (NSHFC>0)){
    ADGSCS=(( ((TWSHS + TWSSS) - (TotWeanWtH + TotWeanWtS)) /(NSHFC+NSSTC) )/DRET);
} else{
    ADGSCS = 0;
}
outfile << "Farm: 
    " << FARM << endl;
outfile << "Address:  
    " << ADDR << endl;
outfile<<endl;
outfile << "  Farm Owner: 
    " << FOWN << endl;
outfile << "  Farm Manager: 
    " << FMAN << endl;
outfile << "  Livestock Manager: 
    " << LMAN << endl;
outfile << endl;
outfile << "  Farm Size: " <<FARMSIZE<< " Acres "

outfile << " Maximum Herd Size w/Replacements: "
<<COWS<<endl;
if ((SpringCalfSeasn==1) && (NumBreedSeasns==1)){
  outfile << " Calving Season: " << "Spring" <<endl;
} else if ((FallCalfSeasn==1) && (NumBreedSeasns==1)){
  outfile<< " Calving Season: " << "Fall" <<endl;
} else if (NumBreedSeasns==2){
  outfile<< " Calving Season: " << "Spring && Fall" <<endl;
}
outfile << endl;
outfile << "Year End Statements for Year: " << YEAR << endl;
outfile << endl;
outfile.setf(ios::fixed,iOS::floatfield);
outfile.setf(iOS::showpoint);
outfile<<setprecision(2);
outfile << " PROJECTED INCOME/LOSS STATEMENT " << endl;
outfile << " FOR" << endl;
outfile << " THE COW-CALF OPERATION " << endl;
outfile << "_______________________________ " << endl;
outfile << endl;
outfile << "Projected Income: " << endl;
outfile << endl;
if (TSTR!=0){
  outfile << "Total Steer Revenue.................." <<(double) TTSTR << endl;
  outfile << endl;
} if (THFR!=0){
  outfile << "Total Heifer Revenue...................." <<TTHFR << endl;
  outfile << endl;
} if (TOCR>0){
  outfile << "Total Orphan Calf Revenue............." <<TTOCR << endl;
  outfile << endl;
} if (TCCR>0){
  outfile << "Total Culled Cow Revenue.............." <<TTCCR << endl;
  outfile << endl;
} if (HEDG1>0){
  if (HBIC>0){
    outfile << "Hedging Revenue......................" <<HBIC << endl;
    outfile << endl;
  }
}
else{
outfile << "Hedging Losses......................" << HBIC << endl;
outfile << endl;
}
}
outfile << "Total Projected Gross Revenue......................" << TPIC << endl;
outfile << endl;
outfile << "Projected Expenses:" << endl;
outfile << endl;
if (CostofReplacements>0)
    outfile << "Livestock Purchases............" << CostofReplacements << endl;
if (TPRD>0)
    outfile << "Pasture Rental................" << TPRD << endl;
if (TPMN>0)
    outfile << "Pasture Maintenance................" << TPMN << endl;
if (FeedCost!=0)
    outfile << "Supplements (at net)............" << FeedCost << endl;
if (SMFC>0)
    outfile << "Salt and Minerals............." << SMFC << endl;
if (HDCE>0)
    outfile << "Hedging Expenses..............." << HDCE << endl;
if (IMPC>0)
    outfile << "Implants......................" << IMPC << endl;
if (VMDC>0)
    outfile << "Veterinary Service & Medicine." << VMDC << endl;
if (LSUP>0)
    outfile << "Livestock Supplies............." << LSUP << endl;
if (BDEP>0)
    outfile << "Breeding Bull Expenses........" << BDEP << endl;
if (TAIC>0)
    outfile << "Artificial Insemination........" << TAIC << endl;
if ( (TPCA>0) || (TPCC>0) )
    outfile << "Marketing Transportation Charges:" << endl;
if (TPCA>0)
    outfile << "  Calves......................" << TPCA << endl;
if (TPCC>0)
    outfile << endl;
outfile << " Culled Cows...................." << TPCC << endl;
}

if ( (TFECA>0) || (YDCA>0) || (TFECAC>0) || (YDCAC>0) ) {
    outfile << "Marketing Expenses:" << endl;
    if ( (TFECA>0) || (YDCA>0) ) {
        outfile << "   Calves:" << endl;
        outfile << "     Commissions & Fees...." << TFECA << endl;
        outfile << "     Yardage..............." << YDCA << endl;
    }
    if ( (TFECAC>0) || (YDCAC>0) ) {
        outfile << "   Culled Cows:" << endl;
        outfile << "     Commissions & Fees ...." << TFECA << endl;
        outfile << "     Yardage..............." << YDCAC << endl;
    }

    if (BFRP>0) {
        outfile << "Building & Fence Repairs......" << BFRP << endl;
    }
    if (UTIL>0) {
        outfile << "Utilities....................." << UTIL << endl;
    }
    if (LABO>0) {
        outfile << "Labor.........................." << LABO << endl;
    }
    if (PRAX>0) {
        outfile << "Payroll Taxes...................." << PRAX << endl;
    }
    if (MECO>0) {
        outfile << "Machinery Expenses................" << MECO << endl;
    }
    outfile << endl;
    outfile << "Total Operating Expenses...................." << TOEX << endl;
    outfile << "________________________" << endl;
    outfile << endl;
}

if (INTE>0) {
    outfile << "Interest Expenses........................" << INTE << endl;
    outfile << "________________________" << endl;
}

outfile << "Total Projected Expenses........................" << TECCE << endl;
outfile << "________________________" << endl;
}

if ( (TPIC>0) || (TECCE>0) ) {


if (NETIC >= 0 ){
    outfile << "Net Revenue............................................." << NETIC << endl;
    outfile << "============" << endl;
} else{
    outfile << "Net Loss........................................" << NETIC << endl;
    outfile << "============" << endl;
}
outfile << endl;
outfile << endl;
outfile << endl;
outfile << endl;

outfile << "Statistics of the Cow-Calf Operation" << endl;
outfile << "_______________________________" << endl;
outfile << endl;

outfile << "ANNUAL BIRTH STATISTICS:" <<endl;
outfile << "Number of Calves Born: " << CNUMB << endl;
outfile << "Average Birthweight of Steers : "
    <<(int) AVGBWTS << endl;
outfile << "Average Birthweight of Heifers : "
    <<(int) AVGBWTH << endl;
outfile << "Average Birthweight of all Calves: "
    <<(int) AVGBWT << endl;
outfile << endl;

outfile << "ANNUAL CULLING STATISTICS:" <<endl;
outfile << "Number of Culled Cows Sold: " <<(int) NCCS <<endl;
outfile << "Probably Pregnant : " <<(int) PregCullCow <<endl;
outfile << "Probably Nonpregnant "
    << (int) (NCCS - PregCullCow)<<endl;
outfile << "Aged Culls Sold " << (int) AgedCulls <<endl;
if (CulledCows<0){
    outfile << "Cull Cows On Hand: " <<-CulledCows << endl;
} else{
    outfile << "Cull Cows On Hand: " <<CulledCows << endl;
}
outfile << "Total Weight of Culled Cows Sold: "
<< (int) TWCC << endl;
outfile << "Average Weight of Culled Cows Sold: "
<< (int) AWCC << endl;
outfile << "Culling Rate: " << CULRAT << " %" << endl;
outfile << endl;
outfile << "ANNUAL FEED STATISTICS: " << endl;
outfile << endl;
outfile << "Estimated Forage Available for Grazing: 
((CarryOver * 2.20459) + (TTGRO)) / 2000" << " Tons" << endl;
outfile << "Forage Consumed, Cow-Calf Ops: 
((TPDMIFA*2.20459)/2000)" << " Tons" << endl;
outfile << "Hay Grown on the Farm: "
(hayAVAIL/2000) << " Tons" << endl;
outfile << "Hay Consumption: "
(hayConsump/2000) << " Tons" << endl;
outfile << "Concentrates Consumed: "
(Concentrate/100) << " Cwt." << endl;
outfile << endl;
outfile << "ANNUAL MORTALITY STATISTICS: " << endl;
outfile << "Number of Dead Heifers: " << DNUMH << endl;
outfile << "Number of Dead Steers: " << DNUMS << endl;
outfile << "Number of Dead Calves: " << DNUM << endl;
outfile << "Death Losses of Breeding Females: " << DNUMF << endl;
outfile << endl;
outfile << "Death Rate, Cows: 
(DRAT*100) << " %" << endl;
outfile << "Death Rate, Calves: 
(DEADC*100) << " %" << endl;
outfile << endl;
outfile << "ANNUAL PRODUCTION STATISTICS: " << endl;
outfile << "Number of Orphan Calves Sold: " << NOC << endl;
outfile << "Number of Steer Calves Sold: " << NSTC << endl;
outfile << "Number of Heifer Calves Sold: " << NHFC << endl;
outfile << "Total Number of Weanling Calves Sold: " << (int) TNCS << endl;
outfile << "Total Weight of Steer Calves: " << (int) TWSS << " LBS" << endl;
outfile << "Total Weight of Heifer Calves: "
<< (int) TWHS << " LBS" << endl;
outfile << "Total Weight of All Calves: "
<< (int) TWCS << " LBS" << endl;
outfile << "Average Weight of Steer Calves: " << (int) AWSS << " LBS" << endl;
outfile << "Average Weight of Heifer Calves: " << (int) AWHS << " LBS" << endl;
outfile << "Average Weight of Weanling Calves: " << (int) (TWCS/TNCS) << " LBS" << endl;
outfile << endl;
if (AVFWEANAGE>0){
  if (((AWSS - AVGBWTS)/AVFWEANAGE) > 0) {
    outfile << "Average Daily Gain, Fall Steers: " << ((AWSS - AVGBWTS)/AVFWEANAGE) << " LBS/DAY" << endl;
  }
  if (((AWHS - AVGBWTH)/AVFWEANAGE) > 0) {
    outfile << "Average Daily Gain, Fall Heifers: " << ((AWHS - AVGBWTH)/AVFWEANAGE) << " LBS/DAY" << endl;
  }
  outfile << "Average Daily Gain, Fall Calves: " << (((TWCS/TNCS) - AVGBWT)/AVFWEANAGE) << " LBS/DAY" << endl;
}
if (AVSWEANAGE>0){
  if (((AWSS - AVGBWTS)/AVSWEANAGE) > 0) {
    outfile << "Average Daily Gain, Spring Steers: " << ((AWSS - AVGBWTS)/AVSWEANAGE) << " LBS/DAY" << endl;
  }
  if (((AWHS - AVGBWTH)/AVSWEANAGE) > 0) {
    outfile << "Average Daily Gain, Spring Heifers: " << ((AWHS - AVGBWTH)/AVSWEANAGE) << " LBS/DAY" << endl;
  }
  outfile << "Average Daily Gain, Spring Calves: " << (((TWCS/TNCS) - AVGBWT)/AVSWEANAGE) << " LBS/DAY" << endl;
}
outfile << endl;
if (AVFWEANAGE>0){
  outfile << "Average Weaning Age, Fall Calves: " << (int) AVFWEANAGE << " DAYS" << endl;
}
if (AVSWEANAGE>0){
  outfile << "Average Weaning Age, Spring Calves: " << (int) AVSWEANAGE << " DAYS" << endl;
}
outfile << "ANNUAL REPRODUCTION STATISTICS: " << endl;
outfile << endl;
if (SpringCalfSeasn==1){
  outfile << "Length of the Breeding Season for Spring Calving: " << SEAS << " Days" << endl;
}
else if (FallCalfSeasn==1){
  outfile << "Length of the Breeding Season for Fall Calving: " << endl;
}
" <<SEAS2" Days" <<endl;
outfile <<endl;
}
outfile << "At the End of the Last Breeding Season: " <<endl;
outfile << endl;
if ((SpringCalfSeasn==1) && (NumBreedSeasns==1)){
  outfile << "Total Number of Cycling Breeding Females :
  " <<CowsElig<<endl;
outfile << "Number of Cycling Breeding Females Open : 
  " <<EligCowsOpen<<endl;
outfile << "Replacement Females Less than 13 Months Old :
  " <<CowsInElig<<endl;
}
else if ((FallCalfSeasn == 1) && (NumBreedSeasns<=2) ){
  outfile << "Total Number of Cycling Breeding Females :
  " <<CowsElig1<<endl;
outfile << "Number of Cycling Breeding Females Open :
  " <<EligCowsOpen1<<endl;
outfile << "Replacement Females Less than 13 Months Old :
  " <<CowsInElig1<<endl;
}
outfile << endl;
outfile << "At Current Year-End:" <<endl;
outfile << endl;
outfile << "Bulls on Hand: " <<NBULLS<< endl;
outfile << "Size of the Breeding Herd, Including 
  Replacements: " << (NumCowsYend) <<endl;
outfile << endl;
outfile << "Pregnant Females on Hand: " << PNUM <<endl;
outfile << "Number More than 45 Days Pregnant, Aborting :
  " <<CowsAbort <<endl;
outfile << "Number of Females Failing to Calve: " <<Cows-
  Fail<<endl;
outfile << "Number of Females Failing to Wean a Calf: 
  " <<CowsFailWean<<endl;
outfile << endl;
outfile << "Number of Animals Experiencing Dystocia: 
  " <<endl;
outfile << "2 Year Olds: " << Dystocia2<<endl;
outfile << "3 Year Olds: " << Dystocia3<<endl;
outfile << "4 Year Olds & Older: " << Dystocia4<<endl;
outfile << endl;
outfile << "Number of Bred Heifers Purchased: 
  " <<PHEF<<endl;
outfile << "Replacement Open Heifers Purchased: 
  " <<POHEF<<endl;
outfile << "Number of Within Herd Replacements Selected : 
  " << SELEC <<endl;
outfile << endl;
outfile << "Average Length of Gestation: " << AvGestLen << endl;
outfile << "Average No. Services/Conception" << AvServConcp << endl;
outfile << "Pregnancy Rate: " << CNRAT << " %" << endl;
outfile << "Calving Rate: " << CRATE << " %" << endl;
outfile << "Weaning Rate: " << WRATE << " %" << endl;
outfile << endl;
if (SpringCalfSeasn==1){
    outfile << "Average Calving Date, Spring Calving:"
    << endl;
    outfile << "Day of the Year: Day " << ACDAT << endl;
    outfile << "Days Since the Start of the Breeding Season:"
    << endl;
    outfile << "Heifer Breeding Season: "
    << ((365 - FDBHS) + ACDAT) << " Days " << endl;
    outfile << "Regular Breeding Season: 
    " << ((365 - FDBS) + ACDAT) << " Days " << endl;
    outfile << endl;
    outfile << "Number of Cycling Heifers, Start of Breeding
    Season: " << NumHeifers << endl;
    outfile << endl;
    outfile << "Heifers Cycling During Heifer Breeding Season,
    Spring Calving:"
    << endl;
    outfile << "During the First Third: " << CyclHeifer1 << endl;
    outfile << "During the Second Third: " << CyclHeifer2 << endl;
    outfile << "During the Last Third: " << CyclHeifer3 << endl;
    outfile << "Heifers Pregnant During Heifer Breeding Season,
    Spring Calving:"
    << endl;
    outfile << "During the First Third: " << PregHef1 << endl;
    outfile << "During the Second Third: " << PregHef2 << endl;
    outfile << "During the Last Third: " << PregHef3 << endl;
    outfile << endl;
    outfile << "Total Number of Cows: " << NumCows << endl;
    outfile << endl;
    outfile << "Cows Cycling During Regular Breeding Season,
    Spring Calving:"
    << endl;
    outfile << "During the First Third: " << CyclCow1 << endl;
    outfile << "During the Second Third: " << CyclCow2 << endl;
    outfile << "During the Last Third: " << CyclCow3 << endl;
    outfile << "Cows Pregnant During Regular Breeding Season,
    Spring Calving:"
    << endl;
    outfile << "During the First Third: " << PregCow1 << endl;
    outfile << "During the Second Third: " << PregCow2 << endl;
    outfile << "During the Last Third: " << PregCow3 << endl;
    outfile << endl;
    outfile << "Number of Heifers and Cows Estimated to be "
    << endl;
    outfile << "Chronically Unable to Maintain a Pregnancy "
    << Sterile << endl;
} else if (FallCalfSeasn==1){
outfile << "Average Calving Date, Fall Calving:" << endl;
outfile << "Day of the Year: Day " << ACDAT << endl;
outfile << "Days Since the Start of the Breeding Season:" << endl;
outfile << "Heifer Breeding Season: "
<< ((365 - FDBHS2) + ACDAT) << " Days " << endl;
outfile << "Regular Breeding Season: "
<< ((365 - FDBS2) + ACDAT) << " Days " << endl;
outfile << endl;
outfile << "Number of Cycling Heifers, Start of Breeding Season: " << NumHeifers << endl;
outfile << endl;
outfile << "Heifers Cycling During Heifer Breeding Season, Fall Calving:" << endl;
if (CyclHeifer21>0){
  outfile << "During the First Third: " << CyclHeifer21 << endl;
}
if (CyclHeifer22>0){
  outfile << "During the Second Third: " << CyclHeifer22 << endl;
}
if (CyclHeifer23>0){
  outfile << "During the Last Third: " << CyclHeifer23 << endl;
}
outfile << "Heifers Pregnant During Heifer Breeding Season, Fall Calving:" << endl;
if (PregHef21>0){
  outfile << "During the First Third: " << PregHef21 << endl;
}
if (PregHef22>0){
  outfile << "During the Second Third: " << PregHef22 << endl;
}
if (PregHef23>0){
  outfile << "During the Last Third: " << PregHef23 << endl;
}
outfile << endl;
outfile << "Total Number of Cows: " << NumCows << endl;
outfile << endl;
outfile << "Cows Cycling During Regular Breeding Season, Fall Calving:" << endl;
if (CyclCow21>0){
  outfile << "During the First Third: " << CyclCow21 << endl;
}
if (CyclCow22>0){
  outfile << "During the Second Third: " << CyclCow22 << endl;
}
if (CyclCow23>0){
  outfile << "During the Last Third: " << CyclCow23 << endl;
outfile << "Cows Pregnant During Regular Breeding Season, Fall Calving:" << endl;
if (PregCow21>0){
    outfile << "During the First Third: " << PregCow21 << endl;
}
if (PregCow22>0){
    outfile << "During the Second Third: " << PregCow22 << endl;
}
if (PregCow23>0){
    outfile << "During the Last Third: " << PregCow23 << endl;
}
outfile << endl;
outfile << "Number of Heifers and Cows Estimated to be Chronically Unable to Maintain a Pregnancy" << Sterile << endl;
outfile << endl;
outfile << endl;
outfile << "BODY CONDITION SCORES, OPEN & CYCLING COWS & REPLACEMENT" << endl;
outfile << "HEIFERS AT THE END OF THE LAST BREEDING SEASON:" << endl;
outfile << endl;
if (BCScore[1]>0){
    outfile << "Breeding Females, BCS of 1: " << BCScore[1] << endl;
}
if (BCScore[2]>0){
    outfile << "Breeding Females, BCS of 2: " << BCScore[2] << endl;
}
if (BCScore[3]>0){
}
if (BCScore[4]>0){
    outfile << "Breeding Females, BCS of 4: " << BCScore[4] << endl;
}
if (BCScore[5]>0){
    outfile << "Breeding Females, BCS of 5: " << BCScore[5] << endl;
}
if (BCScore[6]>0){
    outfile << "Breeding Females, BCS of 6: " << BCScore[6] << endl;
}
if (BCScore[7]>0){
    outfile << "Breeding Females, BCS of 7: " << BCScore[7] << endl;
if (BCScore[8]>0){
    outfile << "Breeding Females, BCS of 8: "
    << BCScore[8] << endl;
}

if (BCScore[9]>0){
    outfile << "Breeding Females, BCS of 9: "
    << BCScore[9] << endl;
}

outfile << "AVERAGE AGE OF THE MATURE BREEDING HERD: "
    "<<AVEAGE" << " Years" <<endl;

outfile << "AGE DISTRIBUTION OF THE BREEDING HERD: "
    "<<endl;

if (AgeBreeders[0]>0){
    outfile << "Breeding Females on Hand, Less than 1 Year :" 
    "<< AgeBreeders[0] <<endl;
}

if (AgeBreeders[1]>0){
    outfile << "Breeding Females on Hand, Yearlings : " 
    "<< AgeBreeders[1] <<endl;
}

if (AgeBreeders[2]>0){
    outfile << "Breeding Females on Hand, 2 Year Olds :" 
    "<< AgeBreeders[2] <<endl;
}

if (AgeBreeders[3]>0){
    outfile << "Breeding Females on Hand, 3 Year Olds : " 
    "<< AgeBreeders[3] <<endl;
}

if (AgeBreeders[4]>0){
    outfile << "Breeding Females on Hand, 4 Year Olds :" 
    "<< AgeBreeders[4] <<endl;
}

if (AgeBreeders[5]>0){
    outfile << "Breeding Females on Hand, 5 Year Olds :" 
    "<< AgeBreeders[5] <<endl;
}

if (AgeBreeders[6]>0){
    outfile << "Breeding Females on Hand, 6 Year Olds :" 
    "<< AgeBreeders[6] <<endl;
}

if (AgeBreeders[7]>0){
    outfile << "Breeding Females on Hand, 7 Year Olds :" 
    "<< AgeBreeders[7] <<endl;
}

if (AgeBreeders[8]>0){
    outfile << "Breeding Females on Hand, 8 Year Olds :" 
    "<< AgeBreeders[8] <<endl;
if (AgeBreeders[9]>0){
    outfile << " Breeding Females on Hand, 9 Year Olds :
    " << AgeBreeders[9] << endl;
}
if (AgeBreeders[10]>0){
    outfile << " Breeding Females on Hand, 10 Year Olds :
    " << AgeBreeders[10] << endl;
}
if (AgeBreeders[11]>0){
    outfile << " Breeding Females on Hand, 11 Year Olds :
}
if (AgeBreeders[12]>0){
    outfile << " Breeding Females on Hand, 12 Year Olds :
    " << AgeBreeders[12] << endl;
}
if (AgeBreeders[13]>0){
    outfile << " Breeding Females on Hand, 13 Year Olds :
    " << AgeBreeders[13] << endl;
}
if (AgeBreeders[14]>0){
    outfile << " Breeding Females on Hand, 14 Year Olds +:
    " << AgeBreeders[14] << endl;
    outfile<<endl;
    outfile<<endl;
    outfile << " PROJECTED PARTIAL BUDGET " << endl;
    outfile <<"FOR" << endl;
    outfile <<"THE STOCKER CALF OPERATION" << endl;
    outfile <<"__________________________" << endl;
    outfile<<endl;
    outfile<<"Additional Projected Income (Loss):"<<endl;
    outfile<<endl;
    if (TSSTR>0){
        outfile << "Total Stocker Steer Revenue........." <<TTSSTR
        << endl;
        outfile << endl;
    }
    if (TSHFR>0){
        outfile << "Total Stocker Heifer Revenue........." <<TTSHFR
        << endl;
        outfile << endl;
    }
    if (HEDG2!=0){
        if (HBIS > 0){
            outfile << "Hedging Revenue.................." <<HBIS <<
            endl;
            outfile<<endl;
        }
    else{
outfile << "Hedging Losses....................." <<HBIS << endl;
outfile<<endl;
}
}
outfile << "Total Additional Projected Revenue.............." <<TAPIS << endl;
outfile << "_____________________________" << endl;
outfile<<endl;
outfile << "Additional Projected Expenses:" << endl;
outfile<<endl;
if (NumBreedSeasns==1){
  if ((SpringCalfSeasn==1) && (DRET + WEANSDATE>365)){
    if (YEAR==1) {
      outfile << "Total Cost of Stocker Steers Sold....." <<(ATTSTR0 * (NSSTC + DNUMSS))<< endl;
      outfile << "Total Cost of Stocker Heifers Sold...." <<(ATTHFR0 * (NSHFC + DNUMHS))<< endl;
    }
    if (YEAR>1){
      outfile << "Total Cost of Stocker Steers Sold....." <<TTSTR2 << endl;
      outfile << "Total Cost of Stocker Heifers Sold...." <<TTHFR2 << endl;
    }
  } else if ((SpringCalfSeasn==1) && (DRET + WEANSDATE<=365)){
    outfile << "Total Cost of Stocker Steers Sold....." <<TTSTR << endl;
    outfile << "Total Cost of Stocker Heifers Sold...." <<TTHFR << endl;
  }
  else if ((FallCalfSeasn==1) && (DRET +WEANFDATE>365)){
    if (YEAR==1) {
      outfile << "Total Cost of Stocker Steers Sold....." <<(ATTSTR0 * (NSSTC + DNUMSS))<< endl;
      outfile << "Total Cost of Stocker Heifers Sold...." <<(ATTHFR0 * (NSHFC + DNUMHS))<< endl;
    }
    if (YEAR>1){
      outfile << "Total Cost of Stocker Steers Sold....." <<TTSTR2 << endl;
      outfile << "Total Cost of Stocker Heifers Sold...." <<TTHFR2 << endl;
    }
  } else if ((FallCalfSeasn==1) && (DRET +WEANFDATE<=365)){
    outfile << "Total Cost of Stocker Steers Sold....." <<TTSTR << endl;
    outfile << "Total Cost of Stocker Heifers Sold...." <<TTHFR
<< endl;
}
}
else if (NumBreedSeasns==2){
if (((SpringCalfSeasn==1) && (DRET + WEANSDATE>365))
&& ((FallCalfSeasn==1) && (DRET +WEANFDATE>365))){
if (YEAR==1) {
outfile << "Total Cost of Stocker Steers Sold...." << ATTSTR0 * (NSSTC + DNUMSS) << endl;
outfile << "Total Cost of Stocker Heifers Sold...." << (ATTHFR0 * (NSHFC + DNUMHS))<< endl;
}
if (YEAR>1){
outfile << "Total Cost of Stocker Steers Sold...." <<TTSTR2 << endl;
outfile << "Total Cost of Stocker Heifers Sold...." <<TTHFR2 << endl;
}
else if (((SpringCalfSeasn==1) && (DRET + WEANSDATE<=365))
&& ((FallCalfSeasn==1) && (DRET +WEANFDATE<=365))){
outfile << "Total Cost of Stocker Steers Sold...." <<TTSTR << endl;
outfile << "Total Cost of Stocker Heifers Sold...." <<TTHFR << endl;
}
}
else if (((SpringCalfSeasn==1) && (DRET + WEANSDATE>365)) &&
((FallCalfSeasn==1) && (DRET +WEANFDATE<=365))){
if (YEAR==1){
outfile << "Total Cost of Stocker Steers Sold...." <<((ATTSTR0 * (NSSTC + DNUMSS)) * (SSCALF/(SSCALF+FSCALF)))
+ (TTSTR * (FSCALF/(SSCALF+FSCALF))))<< endl;
outfile << "Total Cost of Stocker Heifers Sold...." <<((ATTHFR0 * (NSHFC + DNUMHS)) *(SSCALF/(SSCALF+FSCALF)))
+ (TTHFR * (FSCALF/(SSCALF+FSCALF)))) << endl;
else if (YEAR>1){
outfile << "Total Cost of Stocker Steers Sold...." <<((TTSTR2 * (SSCALF/(SSCALF+FSCALF))) + (TTSTR *
FSCALF/(SSCALF+FSCALF))))<< endl;
outfile << "Total Cost of Stocker Heifers Sold...." <<((TTHFR2 *(SSCALF/(SSCALF+FSCALF))) + (TTHFR *
FSCALF/(SSCALF+FSCALF)))) << endl;
}
else if (((SpringCalfSeasn==1) && (DRET + WEANSDATE<365)) &&
((FallCalfSeasn==1) && (DRET +WEANFDATE>365))){
if (YEAR==1){
outfile << "Total Cost of Stocker Steers Sold...." <<((ATTSTR0 * (NSSTC + DNUMSS)) * (FSCALF/(SSCALF+FSCALF)))
+ (TTSTR * (SSCALF/(SSCALF+FSCALF))))<< endl;
}
outfile << "Total Cost of Stocker Heifers Sold...." <<
((((ATTHFR0 * (NSHFC + DNUMHS)) * (FSCALF/(SSCALF+FSCALF)))
+ (TTHFR * (SSCALF/(SSCALF+FSCALF)))) << endl;
else if (YEAR>1){
  outfile << "Total Cost of Stocker Steers Sold....."
  <<((TSTR2 * (FSCALF/(SSCALF+FSCALF))) + (TST * (SSCALF/(SSCALF+FSCALF)))) << endl;
else if (TPRDS>0){
  outfile << "Pasture Rental........................" <<TPRDS << endl;
} else if (TPMNS>0){
  outfile << "Pasture Maintenance...................." <<TPMNS << endl;
} else if (STFeedCost>0){
  outfile << "Supplemental Feeds........................" <<STFeedCost << endl;
} else if (SMFS>0){
  outfile << "Salt and Minerals......................" <<SMFS << endl;
} else if (HDCS>0){
  outfile << "Hedging Expenses........................" <<HDCS << endl;
} else if (Implants>0){
  outfile << "Implants..............................." <<IMPSC << endl;
} else if (Veterinary Service & Medicine........" <<VMDCS << endl;
else if (Livestock Supplies..................." <<LSUPS << endl;
if (TPCSC>0) {
  outfile << "Market Transportation Charges........" <<TPCSC << endl;
} else if (TFECAS>0) || (YDCAS>0) ){
  outfile << "Marketing Expenses:" << endl;
} else if (TFECAS>0){
  outfile << "Commissions & Fees....................." <<TFECAS << endl;
} else if (YDCAS>0){
if (BFRPS>0)
    outfile << "Building & Fence Repairs.............." << BFRPS << endl;
if (UTILS>0)
    outfile << "Utilities............................." << UTILS << endl;
if (LABOS>0)
    outfile << "Labor................................." << LABOS << endl;
if (PRAXS>0)
    outfile << "Payroll Taxes........................." << PRAXS << endl;
if (INSUS>0)
    outfile << "Insurance Expenses...................." << INSUS << endl;
if (MESO>0)
    outfile << "Machinery Expenses...................." << MESO << endl;
outfile << endl;
outfile << "Additional Operating Expenses...........
__________
________________
______________
interest expenses...............................
__________
________________
______________
Total Additional Expenses.......................
__________
________________
______________
Net Additional Revenue.........................." << NETICS << endl;
else
    outfile << "Net Additional Loss............................." << NETICS << endl;
}
outfile << "Stocker Calf Statistics" << endl;
outfile << " ____________________________________" << endl;
outfile << "Number of Stocker Calves on Hand At Year End: " << (int) NumStkrYEnd << endl;
outfile << "ANNUAL FEED STATISTICS: " << endl;
outfile << "Forage Consumed, Stocker Ops: " << ((TSPDMIFA*2.20459)/2000) << " Tons" << endl;
outfile << "Hay Consumed, Stocker Ops: " << ((hayConsumpS)/2000) << " Tons" << endl;
outfile << "Concentrates Consumed, Stocker Ops: " << ((ConcentrateS)/100) << " Cwt." << endl;
outfile << "ANNUAL MORTALITY STATISTICS: " << endl;
outfile << "Number of Dead Stocker Calves: " << (DNUMHS + DNUMSS) << endl;
outfile << "Number of Dead Stocker Heifers: " << DNUMHS << endl;
outfile << "Number of Dead Stocker Steers: " << DNUMSS << endl;
outfile << "ANNUAL PRODUCTION & MARKETING STATISTICS: " << endl;
outfile << "Number of Stocker Calves Sold: " << (NSSTC + NSHFC) << endl;
outfile << "Number of Stocker Steers Sold: " << NSSTC << endl;
outfile << "Number of Stocker Heifers Sold: " << NSHFC << endl;
outfile << "Average Weight of Stocker Calves Sold: " << ((int) ( (TWSHS + TWSSS) / ((double)NSHFC + (double) NSSTC) ) << " LBS" << endl;
outfile << "Average Weight of Stocker Steers Sold: " << (int) AWSSS << " LBS" << endl;
outfile << " Average Weight of Stocker Heifers Sold: " << (int) AWSHS << " LBS"<<endl;
outfile << endl;
if (ADGSCS > 0){
outfile << "Average Daily Gain of Stocker Calves Sold: " << ADGSCS << " LBS"<<endl;
}
if (ADGSS>0){
outfile << " Average Daily Gain of Stocker Steers: " << ADGSS << " LBS"<<endl;
}
if (ADGSH>0){
outfile << " Average Daily Gain of Stocker Heifers: " << ADGSH << " LBS"<<endl;
}
outfile << endl;
if (FSCALF>0){
outfile << "Fall Born Stocker Calves Are Sold On: DAY "
<< int (FSELLDATE/(double)FSCALF)<< endl;
outfile << endl;
}
if (SSCALF>0){
outfile << "Spring Born Stocker Calves Are Sold On: DAY "
<< int (SSELLDATE/(double)SSCALF)<< endl;
outfile << endl;
}
outfile<< endl;
if ((calcuADG==0) && (ADGSCS < antiADG) ){
outfile << "NOTE: Simulation analysis suggests that grass fed stocker"
outfile << " calves grazing the specified forages and fed the "
outfile << " supplemental feeds designated to meet nutritional shortfalls "
outfile << " cannot support the anticipated ADG. "
outfile << endl;
outfile << endl;
}
if (YEAR==1){
outfile << "Average COST of Stocker Calves Sold "<<endl;
outfile << " Includes Death Losses (if any), $/Cwt Sold : "
outfile << "((TTSTR0+TTHFR0)/ (TWSHS + TWSSS)) * 100)"<<endl;
}
else if (YEAR>1){
outfile << "Average COST of Stocker Calves Sold "<<endl;
outfile << " Includes Death Losses (if any), $/Cwt Sold : "
outfile << "((TTSTR2+TTHFR2)/ (TWSHS + TWSSS)) * 100)"<<endl;
}
outfile << "Average Price RECEIVED For Stocker
Calves" << endl;
outfile << "  After Applying Premiums/Discounts, $/Cwt Sold:
    " << (((TTSSTR + TTSHFR)/(TWSHS + TWSSS))* 100) << endl;
outfile << endl;
outfile << "Average Stocker Calf Price REQUIRED to BREAK-EVEN, $/Cwt Sold:
    " << (((TESTE)/(TWSHS + TWSSS))* 100) << endl;
outfile << endl;
outfile << endl;
outfile << "SENSITIVITY ANALYSIS: "
outfile << endl;
outfile << "NOTE: Sensitivity Analysis answers WHAT-IF
Questions Based upon the Information Provided in the Dialog Pages" << endl;
outfile << endl;
outfile << "     Average       Average      Simulated " <<endl;
outfile << "Stocker Calf Cost  Stocker Calf Price     Profit (Loss) " <<endl;
outfile << "     ($/Cwt.)  Received ($/Cwt.)        ($/Cwt.) " <<endl;
outfile << "------------------ --------------------    --------------" <<endl;
outfile << endl;
if (YEAR==1){
    outfile << " << (((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
    (NSHFC+NSSTC))))* 100) - 5.00)<< " << (((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 5.00)" << " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
    DRET)* (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
    outfile << " << (((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
    (NSHFC+NSSTC))))* 100) - 4.00)<< " << (((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 3.00)" << " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
    DRET)* (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
    outfile << " << (((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
    (NSHFC+NSSTC))))* 100) - 2.00)<< " << (((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 1.00)" << " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
    DRET)* (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
    outfile << " << (((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
    (NSHFC+NSSTC))))* 100) - 1.00)<< " << (((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 0.00)" << " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
    DRET)* (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
    outfile << " << (((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
    (NSHFC+NSSTC))))* 100) - 0.00)<< " << (((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 0.00)" << " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
    DRET)* (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
TWSSS) * 100) - 0.00) << " " << (((((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 5.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 1.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.00)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 5.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 2.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 5.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 3.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 5.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 4.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 5.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 5.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << endl;
outfile << " " << (((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 4.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 5.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 4.00) << " " << (((((TTSTR + TTHFR)/(TWSHS +
TWSSS) * 100) - 4.00) << endl;
outfile << " <<=(((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) } << endl;
outfile << " <<=(((TTSTR1+TTHFR1)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 4.00) << " <<=(((TTSTR + TTHFR) / (TWSHS + TWSSS)) * 100) - 2.00) << " <<=(((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) } << endl;
outfile << " <<=(((TTSTR1+TTHFR1)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 1.00) << " <<=(((TTSTR + TTHFR) / (TWSHS + TWSSS)) * 100) - 1.00) << " <<=(((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) } << endl;
outfile << " <<=(((TTSTR1+TTHFR1)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 0.00) << " <<=(((TTSTR + TTHFR) / (TWSHS + TWSSS)) * 100) + 1.00) << " <<=(((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) } << endl;
outfile << " <<=(((TTSTR1+TTHFR1)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 2.00) << " <<=(((TTSTR + TTHFR) / (TWSHS + TWSSS)) * 100) + 2.00) << " <<=(((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) } << endl;
outfile << " <<=(((TTSTR1+TTHFR1)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 3.00) << " <<=(((TTSTR + TTHFR) / (TWSHS + TWSSS)) * 100) + 3.00) << " <<=(((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) } << endl;
DRET) * (NSHFC+NSSTC) ) * 0.04) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 4.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 4.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET) * (NSHFC+NSSTC)) ) * 0.04) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 4.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 5.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET) * (NSHFC+NSSTC)) ) * 0.03) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 3.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 5.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET) * (NSHFC+NSSTC)) ) * 0.03) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 3.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 3.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET) * (NSHFC+NSSTC)) ) * 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 3.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 2.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET) * (NSHFC+NSSTC)) ) * 0.03) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 3.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 1.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET)* (NSHFC+NSSTC)) * 0.03) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100)
}
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 3.00)" "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 0.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100)
}
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 1.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100)
}
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 2.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100)
}
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 3.00)" "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 1.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100)
}
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 3.00)" "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 3.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100)
}
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 3.00)" "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 4.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100)
}
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 3.00)" "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 5.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100)
}
outfile << endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 2.00)" "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 5.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)) * 0.02) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 4.00)"<< " "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)))* 0.02) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
} )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 3.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 2.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)))* 0.02) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
} )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 1.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)))* 0.02) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
} )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 0.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)))* 0.02) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100
} )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 0.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)))* 0.02) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
} )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) + 1.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)))* 0.02) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
} )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) + 2.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)))* 0.02) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
} )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 1.00)<< " "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET)* (NSHFC+NSSTC)) * 0.01) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
) <<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 3.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 2.00)<< " "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET)* (NSHFC+NSSTC)) * 0.01) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
) <<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 4.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 1.00)<< " "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET)* (NSHFC+NSSTC)) * 0.01) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
) <<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 5.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 3.00)<< " "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS *
DRET)* (NSHFC+NSSTC)) * 0.01) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
) <<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<((((TTSTR + TTHFR)/ (TWSHS + TWSSS)) * 100) - 1.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.01) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 0.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.00)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 1.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 2.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 3.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 4.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 5.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << endl;
outfile << "  " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 5.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))) * .000) - ((TWSHS + TWSSS) * .05)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << "  " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 4.00)" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 4.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))) * .000) - ((TWSHS + TWSSS) * .04)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << "  " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 3.00)" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 3.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))) * .000) - ((TWSHS + TWSSS) * .03)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << "  " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 2.00)" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 2.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))) * .000) - ((TWSHS + TWSSS) * .02)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << "  " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 1.00)" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 1.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))) * .000) - ((TWSHS + TWSSS) * .01)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << "  " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) - 0.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))) * .000) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << "  " "<<(((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)) * 100) + 1.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))) * .000) + ((TWSHS + TWSSS) * .01)) / (TWSHS + TWSSS)) * 100
) )<<endl;
```cpp
outfile << "" "<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 0.00" "<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 2.00" "<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "" "<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 0.00" "<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 3.00" "<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "" "<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 0.00" "<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 4.00" "<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "" "<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 0.00" "<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 5.00" "<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "" "<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 0.00" "<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 6.00" "<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * 0.06)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "" "<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 0.00" "<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 7.00" "<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * 0.07)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "" "<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 0.00" "<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 8.00" "<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * 0.08)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "" "<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 0.00" "<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 9.00" "<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * 0.09)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
```

outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) - 2.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) - 1.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) - 0.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) + 1.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) + 2.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) + 3.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) + 4.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) + 5.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR1+TTHFR1)/((TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 1.00)" << " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)))* 100) + 6.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.06)) / (TWSHS + TWSSS)) * 100
) )<<endl;
833
((NSHFC+NSSTC)) * 100) + 2.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 1.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.02) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) >> endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) + 2.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 2.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.02) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) >> endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) + 2.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 3.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.02) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) >> endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) + 2.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 4.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.02) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) >> endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) + 2.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 5.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.02) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) >> endl;
outfile << endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 5.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.03) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) >> endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 4.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.03) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) >> endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS))* 100) - 3.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) )<<endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS))* 100) - 2.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ) )<<endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS))* 100) - 1.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) )<<endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS))* 100) + 0.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.00)) / (TWSHS + TWSSS)) * 100) ) )<<endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS))* 100) + 1.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) )<<endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS))* 100) + 2.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ) )<<endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS))* 100) + 3.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) )<<endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS))* 100) + 3.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) )<<endl;
TWSSS) * 100) + 4.00) << " "<<(((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.03) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " "<<((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) + 3.00) << " "<<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) + 5.00) << " "<<(((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.03) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) + 4.00) << " "<<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) - 5.00) << " "<<(((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.04) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) + 4.00) << " "<<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) - 3.00) << " "<<(((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.04) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) + 4.00) << " "<<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) - 2.00) << " "<<(((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.04) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) + 4.00) << " "<<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) - 1.00) << " "<<(((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.04) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " "<<((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) + 4.00) << " "<<(((TTSTR + TTSFR)/(TWSHS +
TWSSS) * 100) - 0.00)<<
      "<<(((((NETICS - (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.00)) / (TWSHS + TWSSS)) * 100 ) )<endl;
outfile << " "<<(((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100 ) + 1.00)<< " "<<(((((NETICS - (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<endl;
outfile << " "<<(((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100 ) + 2.00)<< " "<<(((((NETICS - (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )<endl;
outfile << " "<<(((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100 ) + 3.00)<< " "<<(((((NETICS - (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<endl;
outfile << " "<<(((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100 ) + 2.00)<< " "<<(((((NETICS - (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )<endl;
outfile << " "<<(((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100 ) + 3.00)<< " "<<(((((NETICS - (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )<endl;
outfile << endl;
outfile << " "<<(((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00)<< " "<<(((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100 ) + 5.00)<< " "<<(((((NETICS - (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )<endl;
outfile << " "<<(((((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00)<< " "<<(((((TTSTR + TTSHFR)/(TWSHS +
TWSSS)* 100) - 4.00) "\n" << ((NETICS - ((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) "\n" << (((TTSTR + TSHFR)/(TWSHS + TWSSS)) * 100) - 3.00) "\n" << (((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) "\n" << (((TTSTR + TSHFR)/(TWSHS + TWSSS)) * 100) - 1.00) "\n" << (((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) "\n" << (((TTSTR + TSHFR)/(TWSHS + TWSSS)) * 100) + 2.00) "\n" << (((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR1+TTHFR1)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) "\n" << (((TTSTR + TSHFR)/(TWSHS + TWSSS)) * 100) + 3.00) "\n" << (((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
DRET) * (NSHFC+NSSTC)) * 0.05) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;

if (YEAR==2){
    outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
    (NSHFC+NSSTC))) * 100) - 5.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 5.00)<< " "<<(((NETICS - ((TWSHS + TWSSS)-(ADGSCS *
    DRET) * (NSHFC+NSSTC)) * 0.05) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
}

if (YEAR==2){
    outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
    (NSHFC+NSSTC))) * 100) - 5.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 5.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS *
    DRET) * (NSHFC+NSSTC)) * 0.05) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
}

if (YEAR==2){
    outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
    (NSHFC+NSSTC))) * 100) - 3.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 3.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS *
    DRET) * (NSHFC+NSSTC)) * 0.05) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
}

if (YEAR==2){
    outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
    (NSHFC+NSSTC))) * 100) - 2.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
    TWSSS)) * 100) - 2.00)<< " "<<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS *
    DRET) * (NSHFC+NSSTC)) * 0.05) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
    ) )<<endl;
}
((NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100 - 1.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 - 0.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 1.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 2.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 3.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 4.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 5.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 - 1.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 1.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 2.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 3.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 4.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR+TTHFR2)/(TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 5.00) << " " << (((TTSTR + TTHFR2)/(TWSHS + TWSSS))* 100 + 5.00) << " " << (((NETICS + (((TWSHS + TWSSS) - (ADGSCS * DRET)) * (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << endl;
(NSHFC+NSSTC)) * 100) - 4.00) << " " << (((TTSSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 5.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.04) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 4.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 4.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.04) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 4.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 3.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.04) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 4.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 2.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.04) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 4.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 1.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.04) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 4.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 0.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.04) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 4.00) << " " << (((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 1.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.04) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 4.00) << " " << (((TTSTR + TTSHFR)/(TWSHS +
TWSSS) * 100) + 2.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC)) * 0.04) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 4.00) << " " << (((TTSTR + TTSHFR)/((TWSHS + TWSSS)) * 100 + 3.00) << " " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((TTSTR + TTSHFR)/((TWSHS + TWSSS)) * 100) - 5.00) << " " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((TTSTR + TTSHFR)/((TWSHS + TWSSS)) * 100) - 4.00) << " " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((TTSTR + TTSHFR)/((TWSHS + TWSSS)) * 100) - 3.00) << " " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((TTSTR + TTSHFR)/((TWSHS + TWSSS)) * 100) - 3.00) << " " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((TTSTR + TTSHFR)/((TWSHS + TWSSS)) * 100) - 3.00) << " " << (((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
TWSSS)* 100) - 2.00) << " " << (((((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 1.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 1.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 2.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 3.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 4.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 5.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-( (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) << endl;
DRET) * (NSHFC+NSSTC)) * 0.03) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 2.00]<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS))* 100) - 5.00)<<" "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) *(NSHFC+NSSTC)))* 0.02) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS))* 100) - 4.00)<<" "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) *(NSHFC+NSSTC)))* 0.02) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS))* 100) - 3.00)<<" "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) *(NSHFC+NSSTC)))* 0.02) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS))* 100) - 2.00)<<" "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) *(NSHFC+NSSTC)))* 0.02) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS))* 100) - 1.00)<<" "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) *(NSHFC+NSSTC)))* 0.02) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS))* 100) - 0.00)<<" "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) *(NSHFC+NSSTC)))* 0.02) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS))* 100) + 1.00)<<" "<<(((NETICS + ((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)) * 0.02) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * 
(NSHFC+NSSTC)))* 100) - 2.00)" " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 2.00) <<" " <<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 
(outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 2.00) <<" " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 3.00) <<" " <<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 
(outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 2.00) <<" " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 4.00) <<" " <<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 2.00) <<" " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 5.00) <<" " <<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 
 outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 1.00) <<" " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) - 5.00) <<" " <<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.01) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 
 outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 1.00) <<" " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) - 4.00) <<" " <<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.01) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 
 outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) *
(NSHFC+NSSTC)))* 100) - 1.00) <<" " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) - 3.00) <<" " <<(((NETICS + ((TWSHS + TWSSS)-(ADGSCS *
DRET) * (NSHFC+NSSTC)) * 0.01) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 2.00)"
"<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.01) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 0.00)"
"<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * 0.00)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 1.00)"
"<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 2.00)"
"<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 3.00)"
"<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))))* 100) - 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 4.00)"
"<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*100) - 1.00) << " " << (((TTSTR + TTHFR)/((TWSHS + TWSSS))*100) + 5.00) << " " << (((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*0.01) + ((TWSHS + TWSSS)*0.05)) / (TWSHS + TWSSS)) * 100 ) << endl;
outfile << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*100) - 0.00) << " " << (((TTSTR + TTHFR)/((TWSHS + TWSSS))*100) - 5.00) << " " << (((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*0.00) - ((TWSHS + TWSSS)*0.05)) / (TWSHS + TWSSS)) * 100 ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*100) - 4.00) << " " << (((TTSTR + TTHFR)/((TWSHS + TWSSS))*100) - 4.00) << " " << (((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*0.00) - ((TWSHS + TWSSS)*0.04)) / (TWSHS + TWSSS)) * 100 ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*100) - 3.00) << " " << (((TTSTR + TTHFR)/((TWSHS + TWSSS))*100) - 3.00) << " " << (((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*0.00) - ((TWSHS + TWSSS)*0.03)) / (TWSHS + TWSSS)) * 100 ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*100) - 2.00) << " " << (((TTSTR + TTHFR)/((TWSHS + TWSSS))*100) - 2.00) << " " << (((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*0.00) - ((TWSHS + TWSSS)*0.02)) / (TWSHS + TWSSS)) * 100 ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*100) - 1.00) << " " << (((TTSTR + TTHFR)/((TWSHS + TWSSS))*100) - 1.00) << " " << (((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*0.00) - ((TWSHS + TWSSS)*0.01)) / (TWSHS + TWSSS)) * 100 ) << endl;
outfile << " " << (((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*100) - 0.00) << " " << (((TTSTR + TTHFR)/((TWSHS + TWSSS))*100) - 0.00) << " " << (((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))*0.00) + ((TWSHS + TWSSS)*0.00)) / (TWSHS + TWSSS)) * 100 }
849

```c
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 0.00)" "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 1.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* .00) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 )
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 0.00)" "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 2.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* .00) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 )
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 0.00)" "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 3.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* .00) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 )
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 0.00)" "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 4.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* .00) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 )
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 0.00)" "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 5.00)" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* .00) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 )
outfile << endl;
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) + 1.00)" "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 5.00)" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* .01) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 )
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) + 1.00)" "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 4.00)" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET) * (NSHFC+NSSTC)))* .01) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
```

849
outfile << " " <<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 1.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 4.00 << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 0.01) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 4.00 << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 0.02) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 6.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 3.00 << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 0.02) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 7.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS))* 100) + 2.00 << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 0.02) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 2.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 0.00 << " " <<(((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100 ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 1.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 2.00 << " " <<(((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 3.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 3.00 << " " <<(((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 4.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 4.00 << " " <<(((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 5.00 << " " <<(((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) <<endl;
outfile << endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 3.00 << " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 5.00 << " " <<(((NETICS - (((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) <<endl;
outfile << endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 3.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 4.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 3.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 3.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 2.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 2.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 1.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 1.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 0.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 0.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.00)) / (TWSHS + TWSSS)) * 100 ) endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 1.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 1.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 2.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 2.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 3.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 3.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 4.00<< " " <<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 4.00<<" " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) endl;
(NSHFC+NSSTC)))* 100) + 3.00) << " " <<(((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 3.00) << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS)* 0.03)) / (TWSHS + TWSSS)) * 100) ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 4.00) << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS)* 0.04)) / (TWSHS + TWSSS)) * 100) ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 5.00) << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS)* 0.05)) / (TWSHS + TWSSS)) * 100) ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 4.00) << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS)* 0.04)) / (TWSHS + TWSSS)) * 100) ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 4.00) << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS)* 0.03)) / (TWSHS + TWSSS)) * 100) ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 4.00) << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS)* 0.02)) / (TWSHS + TWSSS)) * 100) ) <<endl;
outfile << " " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00) << " " <<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS)* 0.01)) / (TWSHS + TWSSS)) * 100) ) <<endl;
(NSHFC+NSSTC)) * 100) + 4.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 1.00) << " " << (((NETICS - (((TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 0.04) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2) / (TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 100) + 4.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 0.00) << " " << (((NETICS - (((TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2) / (TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 100) + 1.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 2.00) << " " << (((NETICS - (((TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2) / (TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 100) + 3.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 3.00) << " " << (((NETICS - (((TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2) / (TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 100) + 4.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 4.00) << " " << (((NETICS - (((TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((TTSTR2+TTHFR2) / (TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 5.00) << " " << (((NETICS - (((TWSHS + TWSSS) - (ADGCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << endl;
outfile << endl;
outfile << endl;

(NSHFC+NSSTC)) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 5.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100) ) } <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 4.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 3.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 2.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 1.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 0.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 1.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 2.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 3.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 4.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;
outfile << " " << (((TTSTR2 + TTHFR2) / (TWSHS + TWSSS)) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << " " << (((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 5.00) << " " << (((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100) ) ) <<endl;

856
TWSSS)* 100) + 2.00)" << "(((((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) + 3.00)" << "(((((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) + 4.00)" << "(((((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) + 5.00)" << "(((((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) + 5.00)" << "(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) + 5.00)" << "(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) - 4.00)" << "(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) - 2.00)" << "(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )" << endl;

if (YEAR==3){
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) + 4.00)" << "(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) + 5.00)" << "(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) - 4.00)" << "(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )" << endl;
outfile << " " << "(((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))))* 100) + 5.00)" << "%(N" << "(((((TTSTR + TTSSTR)/(TWSHS + TWSSS)) * 100) - 4.00)" << "(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.05) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )" << endl;

857
(NSHFC+NSSTC)))* 100) - 5.00) << "" << (((((TTSSTR + TTSHFR)/(TWSHS + TWSSS))* 100) + 5.00) << "" << (((((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)))* 100 ) ) << endl;
outfile << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)))* 100) - 5.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 0.04) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)))* 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)))* 100) - 4.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 0.04) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)))* 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)))* 100) - 3.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 0.04) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)))* 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)))* 100) - 2.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 0.04) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)))* 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)))* 100) - 1.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 0.04) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)))* 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)))* 100) - 0.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 0.04) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)))* 100 ) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 100) - 5.00) << " " << (((((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC))) ) * 0.05) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)))* 100 ) ) << endl;
(NSHFC+NSSTC)) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 1.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) << endl;
}
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 2.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 3.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 4.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 4.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) + 5.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 5.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 4.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 4.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) << endl;
outfile << " " << (((((TTSTR2+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) - 3.00) << " " << (((((TTSTR + TTSHFR) / (TWSHS + TWSSS)) * 100) - 4.00) << " " << (((NETICS + (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) << endl;
(NSHFC+NSSTC)))* 100) - 3.00) << " 
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)))* 100) - 3.00) << " 
" << (((((NETICS + ((((TWSHS + TWSSS) - ((ADGSCS * DRET)* (NSHFC+NSSTC))))* 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)))* 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))))* 100) - 3.00) << " 
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)))* 100) - 2.00) << " 
" << (((((NETICS + ((((TWSHS + TWSSS) - ((ADGSCS * DRET)* (NSHFC+NSSTC))))* 0.03) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)))* 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))))* 100) - 1.00) << " 
" << (((((NETICS + ((((TWSHS + TWSSS) - (NSHFC+NSSTC))))* 0.03) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)))* 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))))* 100) - 0.00) << " 
" << (((((NETICS + ((((TWSHS + TWSSS) - (NSHFC+NSSTC))))* 0.03) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)))* 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))))* 100) + 1.00) << " 
" << (((((NETICS + ((((TWSHS + TWSSS) - (NSHFC+NSSTC))))* 0.03) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)))* 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))))* 100) + 2.00) << " 
" << (((((NETICS + ((((TWSHS + TWSSS) - (NSHFC+NSSTC))))* 0.03) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)))* 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))))* 100) + 3.00) << " 
" << (((((NETICS + ((((TWSHS + TWSSS) - (NSHFC+NSSTC))))* 0.03) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)))* 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))))* 100) - 3.00) << " 
" << (((((TTSTR + TTSHFR)/(TWSHS +
TWSSS)* 100) + 4.00) <<" <<(((NETICS + ((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC)))* 100) - 3.00)<< " <<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) + 5.00)<<" <<(((NETICS + ((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.03) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 2.00)<< " <<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) - 5.00)<<" <<(((NETICS + ((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 2.00)<< " <<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) - 3.00)<<" <<(((NETICS + ((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 2.00)<< " <<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) - 3.00)<<" <<(((NETICS + ((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 2.00)<< " <<(((TTSTR + TTSFR)/(TWSHS + TWSSS))* 100) - 1.00)<<" <<(((NETICS + ((TWSHS + TWSSS) - (ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.02) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) <<endl;
outfile << " <<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) - 2.00)<< " <<(((TTSTR + TTSFR)/(TWSHS +
TWSSS)*100) - 0.00) <<" "<<(NETICS+((TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*0.02) + ((TWSHS+TWSSS)*.000)) / (TWSHS+TWSSS)*100 ) )<<endl; outfile << " "<<(TTSTR2+TTHFR2)/(TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC))*100) - 2.00) <<" "<<(TTSTR+TTSHFR)/(TWSHS+TWSSS)*100) + 1.00) <<" ((NETICS+((TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*0.02) + ((TWSHS+TWSSS)*.000)) / (TWSHS+TWSSS)*100 ) )<<endl; outfile << " "<<(TTSTR2+TTHFR2)/(TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*100) - 2.00) <<" "<<(TTSTR+TTSHFR)/(TWSHS+TWSSS)*100) + 2.00) <<" ((NETICS+((TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*0.02) + ((TWSHS+TWSSS)*.000)) / (TWSHS+TWSSS)*100 ) )<<endl; outfile << " "<<(TTSTR2+TTHFR2)/(TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*100) - 2.00) <<" "<<(TTSTR+TTSHFR)/(TWSHS+TWSSS)*100) + 3.00) <<" ((NETICS+((TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*0.02) + ((TWSHS+TWSSS)*.000)) / (TWSHS+TWSSS)*100 ) )<<endl; outfile << " "<<(TTSTR2+TTHFR2)/(TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*100) - 2.00) <<" "<<(TTSTR+TTSHFR)/(TWSHS+TWSSS)*100) + 4.00) <<" ((NETICS+((TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*0.02) + ((TWSHS+TWSSS)*.000)) / (TWSHS+TWSSS)*100 ) )<<endl; outfile << " "<<(TTSTR2+TTHFR2)/(TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*100) - 2.00) <<" "<<(TTSTR+TTSHFR)/(TWSHS+TWSSS)*100) + 5.00) <<" ((NETICS+((TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*0.02) + ((TWSHS+TWSSS)*.000)) / (TWSHS+TWSSS)*100 ) )<<endl; outfile << endl; outfile << " "<<(TTSTR2+TTHFR2)/(TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*100) - 1.00) <<" "<<(TTSTR+TTSHFR)/(TWSHS+TWSSS)*100) - 5.00) <<" ((NETICS+((TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*0.01) - ((TWSHS+TWSSS)*.000)) / (TWSHS+TWSSS)*100 ) )<<endl; outfile << " "<<(TTSTR2+TTHFR2)/(TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*100) - 1.00) <<" "<<(TTSTR+TTSHFR)/(TWSHS+TWSSS)*100) - 6.00) <<" ((NETICS+((TWSHS+TWSSS)-(ADGSCS*DRET)*(NSHFC+NSSTC)))*0.01) - ((TWSHS+TWSSS)*.000)) / (TWSHS+TWSSS)*100 ) )<<endl;
TWSSS) * 100) - 4.00)<<" <<(((((NETICS + (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))) * 0.01) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ))<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) - 1.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 3.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))) * 0.01) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ))<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) - 2.00)<< " "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))) * 0.01) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ))<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) - 1.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) - 0.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100) ))<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) - 1.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 1.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ))<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) - 1.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 2.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ))<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) - 1.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 3.00)<<" "<<(((NETICS + (((TWSHS + TWSSS)-(ADGSCS * DRET)* (NSHFC+NSSTC)))) * 0.01) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ))<<endl;
DRET) * (NSHFC+NSSTC)) * 0.01) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 1.00)"<< ""<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)))* 100) + 4.00)"<< ""<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100)
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 1.00)"<< ""<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)))* 100) + 5.00)"<< ""<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.00) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100)
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 0.00)"<< ""<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)))* 100) - 4.00)"<< ""<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.00) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100)
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 0.00)"<" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)))* 100) - 3.00)"<< ""<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.00) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100)
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 0.00)"<" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)))* 100) - 2.00)"<< ""<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.00) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100)
) )<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) - 0.00)"<" "<<(((TTSTR + TTSHFR)/(TWSHS +
TWSSS)))* 100) - 1.00)"<< ""<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC)) * .000 - ((TWSHS + TWSSS) * .01) / (TWSHS + TWSSS)) * 100
 )))<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 0.00)"" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100
 ))<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 1.00)"" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * .01)) / (TWSHS + TWSSS)) * 100
 ))<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 2.00)"" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * .02)) / (TWSHS + TWSSS)) * 100
 ))<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 3.00)"" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * .03)) / (TWSHS + TWSSS)) * 100
 ))<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 4.00)"" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * .04)) / (TWSHS + TWSSS)) * 100
 ))<<endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) - 0.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 5.00)"" "<<(((NETICS + (((TWSHS + TWSSS)-((ADGSCS *
DRET) * (NSHFC+NSSTC))) * .000) + ((TWSHS + TWSSS) * .05)) / (TWSHS + TWSSS)) * 100
 ))<<endl;
outfile << endl;
outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC))) * 100) + 1.00)<< " "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) - 5.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)) * 0.01) - ((TWSHS + TWSSS) * 0.05) / (TWSHS + TWSSS)) * 100
 ) )<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
 (NSHFC+NSSTC)))* 100) + 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
 TWSSS))* 100) - 4.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-(ADGSCS *
 DRET)* (NSHFC+NSSTC)))* 0.01) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
 ) )<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
 (NSHFC+NSSTC)))* 100) + 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
 TWSSS))* 100) - 3.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-(ADGSCS *
 DRET)* (NSHFC+NSSTC)))* 0.01) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
 ) )<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
 (NSHFC+NSSTC)))* 100) + 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
 TWSSS))* 100) - 2.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-(ADGSCS *
 DRET)* (NSHFC+NSSTC)))* 0.01) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
 ) )<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
 (NSHFC+NSSTC)))* 100) + 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
 TWSSS))* 100) - 1.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-(ADGSCS *
 DRET)* (NSHFC+NSSTC)))* 0.01) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
 ) )<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
 (NSHFC+NSSTC)))* 100) + 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
 TWSSS))* 100) - 0.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-(ADGSCS *
 DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100
 ) )<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
 (NSHFC+NSSTC)))* 100) + 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
 TWSSS))* 100) + 1.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-(ADGSCS *
 DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100
 ) )<<endl;
 outfile << " "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS * DRET)*
 (NSHFC+NSSTC)))* 100) + 1.00)<< " "<<(((TTSTR + TTSHFR)/(TWSHS +
 TWSSS))* 100) + 2.00)"" "<<(((NETICS - (((TWSHS + TWSSS)-(ADGSCS *
 DRET)* (NSHFC+NSSTC)))* 0.01) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100

outfile << endl;
outfile << "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00)<< "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS))* 100) - 5.00)<<" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00)<< "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS))* 100) - 4.00)<<" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00)<< "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS))* 100) - 3.00)<<" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00)<< "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS))* 100) - 2.00)<<" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00)<< "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS))* 100) - 1.00)<<" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00)<< "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS))* 100) - 0.00)<<" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * .000)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << "<<(((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 100) + 3.00)<< "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS))* 100) + 1.00)<<" "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<<endl;

870
outfile << "  "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 3.00<< "  "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 2.00]<<"  "<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100
 ) <<endl;
 outfile << "  "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 3.00<< "  "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 3.00)<<"  "<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
 ) <<endl;
 outfile << "  "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 3.00<< "  "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 4.00)<<"  "<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
 ) <<endl;
 outfile << "  "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 4.00<< "  "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 5.00)<<"  "<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.03) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
 ) <<endl;
 outfile << "  "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 4.00<< "  "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 5.00)<<"  "<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.04) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100
 ) <<endl;
 outfile << "  "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 4.00<< "  "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 4.00)<<"  "<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.04) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100
 ) <<endl;
 outfile << "  "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 4.00<< "  "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 3.00)<<"  "<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.04) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
 ) <<endl;
 outfile << "  "<<(((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-((ADGSCS * DRET)*
(NSHFC+NSSTC)))* 100) + 4.00<< "  "<<(((TTSTR + TTHFR)/(TWSHS +
TWSSS)) * 100) + 3.00)<<"  "<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS *
DRET)* (NSHFC+NSSTC)))* 0.04) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100
 ) <<endl;
outfile << " "<<((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS)) * 100) - 2.00)<< " "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.04) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS)) * 100) - 1.00)<< " "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.04) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS)) * 100) + 0.00)<< " "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.00)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS)) * 100) + 1.00)<< " "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS)) * 100) + 2.00)<< " "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS)) * 100) + 3.00)<< " "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
outfile << " "<<((((TTSTR2+TTHFR2)/((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 100) + 4.00)<< " "<<(((TTSTR + TTHFR)/((TWSHS + TWSSS)) * 100) + 4.00)<< " "<<(((NETICS - (((TWSHS + TWSSS)-((ADGSCS * DRET)* (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) )<<endl;
((NSHFC+NSSTC))* 100) + 4.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 5.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.04) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 5.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 4.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 3.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 2.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 1.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) - 0.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.000)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 0.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) - ((TWSHS + TWSSS) * 0.001)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 1.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.002)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 2.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.003)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 3.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.004)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 4.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.005)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 5.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.006)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
outfile << " 
" << (((((TTSTR+TTHFR2)/(TWSHS + TWSSS))-((ADGSCS * DRET) * (NSHFC+NSSTC))) * 100) + 5.00) << "
" << (((((TTSTR + TTSHFR)/(TWSHS + TWSSS)) * 100) + 6.00) << "
" << (((((NETICS - (((TWSHS + TWSSS) - ((ADGSCS * DRET) * (NSHFC+NSSTC))) * 0.05) + ((TWSHS + TWSSS) * 0.007)) / (TWSHS + TWSSS)) * 100) ) ) << endl;
(NSHFC+NSSTC)))* 100) + 5.00) << " " << (((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 1.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.01)) / (TWSHS + TWSSS)) * 100 ) ) << endl;

outfile << " " << (((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 100) + 5.00) << " " << (((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 2.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.02)) / (TWSHS + TWSSS)) * 100 ) ) << endl;

outfile << " " << (((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 100) + 5.00) << " " << (((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 3.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.03)) / (TWSHS + TWSSS)) * 100 ) ) << endl;

outfile << " " << (((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 100) + 5.00) << " " << (((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 4.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.04)) / (TWSHS + TWSSS)) * 100 ) ) << endl;

outfile << " " << (((TTSTR2+TTHFR2)/(TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 100) + 5.00) << " " << (((TTSTR + TTHFR)/(TWSHS + TWSSS)) * 100) + 5.00) << " " << (((NETICS - (((TWSHS + TWSSS)-(ADGSCS*DRET)* (NSHFC+NSSTC)))* 0.05) + ((TWSHS + TWSSS) * 0.05)) / (TWSHS + TWSSS)) * 100 ) ) << endl;

outfile << endl;
}
}
void Market::PurchaseLivestock(AnimalRecord *animal){
    double PregAdjust=1.12;
    double PurchaseCost=0;
    double PRS1[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double PRS2[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double PRS3[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
    double SPA[13]={0,1.015,1.045,1.054,1.041,1.019,1.011,0.997,1.009,0.968,0.948,0.942,0.950};
    double PRH[11]={0,0,0,0,0.89772,0.90813,0.90537,0.88218,0.86462,0.8352,0.7998};
    double PRS[11]={0,0,0,0,1.107,1.064,1.052,1.000,0.97189,0.9398,0.9000};
    PRS1[1] = (PRSA1 * SPA[1]);
    PRS1[2] = (PRSA1 * SPA[2]);
    PRS1[3] = (PRSA1 * SPA[3]);
    PRS1[4] = (PRSA1 * SPA[4]);
    PRS1[5] = (PRSA1 * SPA[5]);
    PRS1[6] = (PRSA1 * SPA[6]);
    PRS1[7] = (PRSA1 * SPA[7]);
    PRS1[8] = (PRSA1 * SPA[8]);
    PRS1[9] = (PRSA1 * SPA[9]);
    PRS1[10] = (PRSA1 * SPA[10]);
    PRS1[12] = (PRSA1 * SPA[12]);
    PRS2[1] = (PRSA2 * SPA[1]);
    PRS2[2] = (PRSA2 * SPA[2]);
    PRS2[3] = (PRSA2 * SPA[3]);
    PRS2[4] = (PRSA2 * SPA[4]);
    PRS2[5] = (PRSA2 * SPA[5]);
    PRS2[6] = (PRSA2 * SPA[6]);
    PRS2[7] = (PRSA2 * SPA[7]);
    PRS2[8] = (PRSA2 * SPA[8]);
    PRS2[9] = (PRSA2 * SPA[9]);
    PRS2[10] = (PRSA2 * SPA[10]);
    PRS2[12] = (PRSA2 * SPA[12]);
    PRS3[1] = (PRSA3 * SPA[1]);
    PRS3[2] = (PRSA3 * SPA[2]);
    PRS3[3] = (PRSA3 * SPA[3]);
    PRS3[4] = (PRSA3 * SPA[4]);
    PRS3[5] = (PRSA3 * SPA[5]);
    PRS3[6] = (PRSA3 * SPA[6]);
    PRS3[7] = (PRSA3 * SPA[7]);
    PRS3[8] = (PRSA3 * SPA[8]);
    PRS3[9] = (PRSA3 * SPA[9]);
    PRS3[10] = (PRSA3 * SPA[10]);
    PRS3[12] = (PRSA3 * SPA[12]);
}
if (YEAR==1) {
    if ((currentDATE>=1) && (currentDATE<32)) {
        PRIS=PRS1[1];
    } else if ((currentDATE>=32) && (currentDATE<60)) {
        PRIS=PRS1[2];
    } else if ((currentDATE>=60) && (currentDATE<91)) {
        PRIS=PRS1[3];
    } else if ((currentDATE>=91) && (currentDATE<121)) {
        PRIS=PRS1[4];
    } else if ((currentDATE>=121) && (currentDATE<152)) {
        PRIS=PRS1[5];
    } else if ((currentDATE>=152) && (currentDATE<182)) {
        PRIS=PRS1[6];
    } else if ((currentDATE>=182) && (currentDATE<213)) {
        PRIS=PRS1[7];
    } else if ((currentDATE>=213) && (currentDATE<244)) {
        PRIS=PRS1[8];
    } else if ((currentDATE>=244) && (currentDATE<274)) {
        PRIS=PRS1[9];
    } else if ((currentDATE>=274) && (currentDATE<305)) {
        PRIS=PRS1[10];
    } else if ((currentDATE>=305) && (currentDATE<335)) {
        PRIS=PRS1[11];
    } else if ((currentDATE>=335) && (currentDATE<=365)) {
        PRIS=PRS1[12];
    } else if (YEAR==2) {
        if ((currentDATE>=1) && (currentDATE<32)) {
            PRIS=PRS2[1];
        } else if ((currentDATE>=32) && (currentDATE<60)) {
            PRIS=PRS2[2];
        } else if ((currentDATE>=60) && (currentDATE<91)) {
            PRIS=PRS2[3];
        } else if ((currentDATE>=91) && (currentDATE<121)) {
            PRIS=PRS2[4];
        }
else if ((currentDATE>=121) && (currentDATE<152)){
    PRIS=PRS2[5];
}
else if ((currentDATE>=152) && (currentDATE<182)){
    PRIS=PRS2[6];
}
else if ((currentDATE>=182) && (currentDATE<213)){
    PRIS=PRS2[7];
}
else if ((currentDATE>=213) && (currentDATE<244)){
    PRIS=PRS2[8];
}
else if ((currentDATE>=244) && (currentDATE<274)){
    PRIS=PRS2[9];
}
else if ((currentDATE>=274) && (currentDATE<305)){
    PRIS=PRS2[10];
}
else if ((currentDATE>=305) && (currentDATE<335)){
    PRIS=PRS2[11];
}
else if ((currentDATE>=335) && (currentDATE<=365)){
    PRIS=PRS2[12];
}
}
else if (YEAR==3){
    if ((currentDATE>=1) && (currentDATE<32)){
        PRIS=PRS3[1];
    }
    else if ((currentDATE>=32) && (currentDATE<60)){
        PRIS=PRS3[2];
    }
    else if ((currentDATE>=60) && (currentDATE<91)){
        PRIS=PRS3[3];
    }
    else if ((currentDATE>=91) && (currentDATE<121)){
        PRIS=PRS3[4];
    }
    else if ((currentDATE>=121) && (currentDATE<152)){
        PRIS=PRS3[5];
    }
    else if ((currentDATE>=152) && (currentDATE<182)){
        PRIS=PRS3[6];
    }
    else if ((currentDATE>=182) && (currentDATE<213)){
        PRIS=PRS3[7];
    }
    else if ((currentDATE>=213) && (currentDATE<244)){
        PRIS=PRS3[8];
    }
    else if ((currentDATE>=244) && (currentDATE<274)){
        PRIS=PRS3[9];
    }
    else if ((currentDATE>=274) && (currentDATE<305)){
        PRIS=PRS3[10];
    }
    else if ((currentDATE>=305) && (currentDATE<335)){
        PRIS=PRS3[11];
    }
    else if ((currentDATE>=335) && (currentDATE<365)){
        PRIS=PRS3[12];
    }
}
**Code Snippet:**

```c
PRIS = PRS3[9];
} else if ((currentDATE >= 274) && (currentDATE < 305)) {
    PRIS = PRS3[10];
} else if ((currentDATE >= 305) && (currentDATE < 335)) {
    PRIS = PRS3[11];
} else if ((currentDATE >= 335) && (currentDATE <= 365)) {
    PRIS = PRS3[12];
}

if ( (animal->SOUR == 1) && (animal->PREG == 0) ) {
    if ((animal->SEX == 0) || (animal->SEX == 3) || (animal->SEX == 4)) {
        if ((animal->CDWT >= 300) && (animal->CDWT < 400)) {
            PurchaseCost = (PRIS * PRH[4]) / 100 * animal->CDWT;
            TotalPurchaseCost += PurchaseCost;
            animal->SOUR = 0;
        } else if ((animal->CDWT >= 400) && (animal->CDWT < 500)) {
            PurchaseCost = (PRIS * PRH[5]) / 100 * animal->CDWT;
            TotalPurchaseCost += PurchaseCost;
            animal->SOUR = 0;
        } else if ((animal->CDWT >= 500) && (animal->CDWT < 600)) {
            PurchaseCost = (PRIS * PRH[6]) / 100 * animal->CDWT;
            TotalPurchaseCost += PurchaseCost;
            animal->SOUR = 0;
        } else if ((animal->CDWT >= 600) && (animal->CDWT < 700)) {
            PurchaseCost = (PRIS * PRH[7]) / 100 * animal->CDWT;
            TotalPurchaseCost += PurchaseCost;
            animal->SOUR = 0;
        } else if ((animal->CDWT >= 700) && (animal->CDWT < 800)) {
            PurchaseCost = (PRIS * PRH[8]) / 100 * animal->CDWT;
            TotalPurchaseCost += PurchaseCost;
            animal->SOUR = 0;
        } else if ((animal->CDWT >= 800) && (animal->CDWT < 900)) {
            PurchaseCost = (PRIS * PRH[9]) / 100 * animal->CDWT;
            TotalPurchaseCost += PurchaseCost;
            animal->SOUR = 0;
        } else if ((animal->CDWT >= 900)) {
            PurchaseCost = (PRIS * PRH[10]) / 100 * animal->CDWT;
            TotalPurchaseCost += PurchaseCost;
            animal->SOUR = 0;
        }
    }
}
```
else if ((animal->SEX == 2) || (animal->SEX == 5)){
    if ((animal->CDWT >= 300) && (animal->CDWT < 400)){
        PurchaseCost=(PRIS*PRS[4])/100 * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    } else if ((animal->CDWT >= 400) && (animal->CDWT < 500)){
        PurchaseCost=(PRIS*PRS[5])/100 * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    } else if ((animal->CDWT >= 500) && (animal->CDWT < 600)){
        PurchaseCost=(PRIS*PRS[6])/100 * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    } else if ((animal->CDWT >= 600) && (animal->CDWT < 700)){
        PurchaseCost=(PRIS*PRS[7])/100 * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    } else if ((animal->CDWT >= 700) && (animal->CDWT < 800)){
        PurchaseCost=(PRIS*PRS[8])/100 * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    } else if ((animal->CDWT >= 800) && (animal->CDWT < 900)){
        PurchaseCost=(PRIS*PRS[9])/100 * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    } else if ((animal->CDWT >= 900) ){
        PurchaseCost=(PRIS*PRS[10])/100 * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    }
    }
}
else if ( (animal->SOUR == 1) && (animal->SEX == 3) &&
    (animal->PREG == 1) ){
    if ((animal->CDWT >= 300) && (animal->CDWT < 400)){
        PurchaseCost=(PRIS*PRH[4])/100 * PregAdjust * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    } else if ((animal->CDWT >= 400) && (animal->CDWT < 500)){
        PurchaseCost=(PRIS*PRH[5])/100 * PregAdjust * animal->CDWT;
        TotalPurchaseCost+=PurchaseCost;
        animal->SOUR=0;
    } else if ((animal->CDWT >= 500) && (animal->CDWT < 600)){

PurchaseCost=(PRIS*PRH[6])/100 * PregAdjust * animal->CDWT;
TotalPurchaseCost+=PurchaseCost;
animal->SOUR=0;
}
else if ((animal->CDWT >= 600) && (animal->CDWT < 700)) {
   PurchaseCost=(PRIS*PRH[7])/100 * PregAdjust * animal->CDWT;
   TotalPurchaseCost+=PurchaseCost;
   animal->SOUR=0;
}
else if ((animal->CDWT >= 700) && (animal->CDWT < 800)) {
   PurchaseCost=(PRIS*PRH[8])/100 * PregAdjust * animal->CDWT;
   TotalPurchaseCost+=PurchaseCost;
   animal->SOUR=0;
}
else if ((animal->CDWT >= 800) && (animal->CDWT < 900)) {
   PurchaseCost=(PRIS*PRH[9])/100 * PregAdjust * animal->CDWT;
   TotalPurchaseCost+=PurchaseCost;
   animal->SOUR=0;
}
else if ((animal->CDWT >= 900)) {
   PurchaseCost=(PRIS*PRH[10])/100 * PregAdjust * animal->CDWT;
   TotalPurchaseCost+=PurchaseCost;
   animal->SOUR=0;
}
if (currentDATE==365)
   CostofReplacements=TotalPurchaseCost;
}

void Market::ResetMarketVariables()
{
   ASHKC=0;
   ASHKH=0;
   ASHKOC=0;
   ASHKS=0;
   ASHKSH=0;
   ASHKSS=0;
   NCCS=0;
   NHFC=0;
   NOC=0;
   NSHFC=0;
   NSSTC=0;
   NSTC=0;
   SALT=0;
   FSELLDATE=0;
   SSELLDATE=0;
   FSCALF=0;
   SSCALF=0;
   TBDCC=0;
   TBDH=0;
   TBDOC=0;
}
TBDS=0;
TBDSH=0;
TBDDS=0;
TCCR=0;
TDSPC=0;
TDSPH=0;
TDSPOC;
TDSPS=0;
TDSPSH=0;
TDSPSS=0;
TESPH=0;
TESPS=0;
TESPSH=0;
TESPSS=0;
TFECA=0;
TFECAC=0;
TFECAS=0;
THFR=0;
TOCR=0;
TotalPurchaseCost=0;
CostofReplacements=0;
TotWeanWtH=0;
TotWeanWtS=0;
TSHFR=0;
TSHKC=0;
TSHKH=0;
TSHKOC=0;
TSHKS=0;
TSHKSH=0;
TSHKSS=0;
TSTR=0;
TTSTR0=0;
TTSTR1=0;
TTSTR2=0;
TTHFR0=0;
TTHFR1=0;
TTHFR2=0;
TSTR1=0;
TBDS1=0;
TDSPS1=0;

if ((YEAR==3) || (YEAR<0) ) {
    TTSTR1=0;
    TTSTR2=0;
    TTHFR1=0;
    TTHFR2=0;
    TSTR1=0;
    TBDS1=0;
    TDSPS1=0;
TESPS1=0;
THFR1=0;
TBDH1=0;
TDSPH1=0;
TESPH1=0;

}
Nutrition.cpp
#include "stdafx.h"
#include "HerdSim.h"
#include "HerdSimDlg.h"
#include <iostream.h>
#include <string.h>
#include <math.h>
#include "cow.h"
#include "nutrit.h"
#include "tools.h"
#include "forage.h"
extern int YEAR;
double mainSDM;
extern double mainFDM;
double mainSDMC;
extern int COWS;
double AdjFeedCost;
double AdjSTFeedCost;
double hayRevenue;
extern double hayAVAIL;
extern double SELLHAY;
extern double HAYCOST;
double hayVALUE;
double hayCost;
extern int calcuADG;
extern int calcuseasADG;
double SAME;
double SANEG;
double SANEM;
double CSAME;
double CSANEG;
double CSANEM;
double GREDCost;
double CGREDCost;
double TSU;
double DIPM;
double FeedCost;
double STFeedCost;
extern int SREG;
extern int currentDate;
extern int CurrentSIZE;
extern int CurrentSIZES;
extern int TERR;
extern char IONO;
extern double Tc;
extern double Tp;
extern double PAVAIL;
extern double FANEM;
extern double FAME;
extern double FANEG;
extern double TDNP;
extern int IMPH;
extern int IMPS;
extern int IMPSS;
extern int IMPHS;
char SUPPL;
double PDMIF;
double PDMIFA;
double PDMIS;
double CPDMIS;
double SPDMI;
double SPDMIFA;
double SPDMIS;
double TPDMIF;
double TPDMIFA;
double TPDMIS;
double TSPDMIFA;
double TSPDMIS;
double hayConsump;
double Concentrate;
double hayConsumpS;
double ConcentrateS;

Nutrition::Nutrition(){
    int count;

    GRCost[0]=0;
    GRCost[1]=.03;
    GRCost[2]=.03;
    GRCost[3]=.03;
    GRCost[4]=.03;
    GRCost[5]=.03;
    GRCost[6]=.03;
    GRCost[7]=.03;
    GRCost[8]=.03;
    GRCost[9]=.05;
    GRCost[10]=.05;
    GRCost[11]=.05;
    GRCost[12]=.09;
    GRCost[13]=.03;
    GRCost[14]=.04;
    GRCost[15]=.05;
    GRCost[16]=0;
    GRCost[17]=0;
    GRCost[18]=0;
    GRCost[19]=0;
    GRCost[20]=0;
    SDM[1]=0.88;
    SDM[2]=0.89;
SDM[3]=0.884;
SDM[4]=0.89;
SDM[5]=0.896;
SDM[6]=0.886;
SDM[7]=0.9;
SDM[8]=0.91;
SDM[9]=0.905;
SDM[10]=0.87;
SDM[11]=0.43;
SDM[12]=0.89;
SDM[13]=0.88;
SDM[14]=0.88;
SDM[15]=0.89;
SDM[16]=0.0;
SDM[17]=0.0;
SDM[18]=0.0;
SDM[19]=0.0;
SDM[20]=0.0;
SME[1]=2.10;
SME[2]=2.128;
SME[3]=2.19;
SME[4]=2.36;
SME[5]=2.37;
SME[6]=2.25;
SME[7]=2.16;
SME[8]=2.06;
SME[9]=2.38;
SME[10]=3.65;
SME[11]=2.92;
SME[12]=2.90;
SME[13]=1.98;
SME[14]=2.91;
SME[15]=2.94;
SME[16]=0.0;
SME[17]=0.0;
SME[18]=0.0;
SME[19]=0.0;
SME[20]=0.0;
SNEM[0]=0.0;
SNEM[1]=1.24;
SNEM[2]=1.244;
SNEM[3]=1.3;
SNEM[4]=1.22;
SNEM[5]=1.28;
SNEM[6]=1.2;
SNEM[7]=1.31;
SNEM[8]=1.2;
SNEM[9]=1.38;
SNEM[10]=2.24;
SNEM[11]=1.7;
SNEM[12]=1.95;
SNEM[13]=1.33;
SNEM[14]=1.95;
SNEM[15]=1.98;
SNEM[16]=0.0;
SNEM[17]=0.0;
SNEM[18]=0.0;
SNEM[19]=0.0;
SNEM[20]=0.0;
SNEG[0]=0.0;
SNEG[1]=0.68;
SNEG[2]=0.704;
SNEG[3]=0.73;
SNEG[4]=0.66;
SNEG[5]=0.71;
SNEG[6]=0.64;
SNEG[7]=0.73;
SNEG[8]=0.64;
SNEG[9]=0.79;
SNEG[10]=1.55;
SNEG[11]=1.08;
SNEG[12]=1.30;
SNEG[13]=0.75;
SNEG[14]=1.31;
SNEG[15]=1.33;
SNEG[16]=0.0;
SNEG[17]=0.0;
SNEG[18]=0.0;
SNEG[19]=0.0;
SNEG[20]=0.0;
SOPT1="XXX";
SOPT2="XXX";
SOPT3="XXX";
SOPT4="XXX";
SOPT5="XXX";
STDN[0]=0.0;
STDN[1]=0.58;
STDN[2]=0.568;
STDN[3]=0.594;
STDN[4]=0.570;
STDN[5]=0.578;
STDN[6]=0.57;
STDN[7]=0.57;
STDN[8]=0.58;
STDN[9]=0.59;
STDN[10]=0.920;
STDN[11]=0.72;
STDN[12]=0.85;
STDN[13]=0.53;
STDN[14]=0.85;
STDN[15]=0.77;
STDN[16]=0.0;
STDN[17]=0.0;
STDN[18]=0.0;
STDN[19]=0.0;
STDN[20]=0.0;
SU[0]=0.0;
SU[1]=0.5;
SU[2]=0.0;
SU[3]=0.0;
SU[4]=0.25;
SU[5]=0.0;
SU[6]=0.0;
SU[7]=0.0;
SU[8]=0.0;
SU[9]=0.0;
SU[10]=0.0;
SU[11]=0.0;
SU[12]=0.0;
SU[13]=0.25;
SU[14]=0.0;
SU[15]=0.0;
SU[16]=0.0;
SU[17]=0.0;
SU[18]=0.0;
SU[19]=0.0;
SU[20]=0.0;
SUPPL=1;
mainSDM=0;
mainSDMC=0;
APDMIF=0;
ASPDMIF=0;
DIPM=0;
FeedCost=0;
STFeedCost=0;
AdjFeedCost=0;
AdjSTFeedCost=0;
hayRevenue=0;
PDMI=0;
SAME=0;
SANEG=0;
SANEM=0;
CSAME=0;
CSANEG=0;
CSANEM=0;
TPDMIF=0;
TSPDMIF=0;
TPDMIFA=0;
TSPDMIFA=0;
TPDMIS=0;
TCPDMIS=0;
TSPDMIS=0;
GRAZE=0;
GUNIT=0;
PDMIF=0;
PDMIFA=0;
PDMIS=0;
SPDMI=0;
SPDMIF=0;
SPDMIFA=0;
SPDMIS=0;
SUPP=0;
hayConsump=0;
Concentrate=0;
hayConsumpS=0;
ConcentrateS=0;
hayVALUE=(SELLHAY/2000);
hayCost=(HAYCOST/2000);
for (count=0; count<21; count++){
mainSDM += (SDM[count]*SU[count]);
SAME += SME[count]*SU[count];
SANEM += SNEM[count]*SU[count];
SANEG += SNEG[count]*SU[count];
}
for (count=0; count<11; count++){
mainSDMC += (SDM[count] *
+SU[8]+SU[9]));
CSAME += (SME[count] *
+SU[8]+SU[9]));
CSANEM += (SNEM[count] *
+SU[8]+SU[9]));
CSANEG += (SNEG[count] *
+SU[8]+SU[9]));
}

void Nutrition::UseInput7(){
int response;
double TSU;
do{
CInput7* input=new CInput7;
input->m_grcost1=GRCost[1];
input->m_grcost2=GRCost[2];
input->m_grcost3=GRCost[3];
input->m_grcost4=GRCost[4];
input->m_grcost5=GRCost[5];
input->m_grcost6=GRCost[6];
input->m_grcost7=GRCost[7];
input->m_grcost8=GRCost[8];
input->m_grcost9=GRCost[9];
input->m_grcost10=GRCost[10];
input->m_grcost11=GRCost[11];
input->m_grcost12=GRCost[12];
input->m_grcost13=GRCost[13];
input->m_grcost14=GRCost[14];
input->m_grcost15=GRCost[15];
input->m_grcost16=GRCost[16];
input->m_grcost17=GRCost[17];
input->m_grcost18=GRCost[18];
input->m_grcost19=GRCost[19];
input->m_grcost20=GRCost[20];
input->m_sopt1=SOPT1;
input->m_sopt2=SOPT2;
input->m_sopt3=SOPT3;
input->m_sopt4=SOPT4;
input->m_sopt5=SOPT5;
input->m_su1=SU[1];
input->m_su2=SU[2];
input->m_su3=SU[3];
input->m_su4=SU[4];
input->m_su5=SU[5];
input->m_su6=SU[6];
input->m_su7=SU[7];
input->m_su8=SU[8];
input->m_su9=SU[9];
input->m_su10=SU[10];
input->m_su11=SU[11];
input->m_su12=SU[12];
input->m_su13=SU[13];
input->m_su14=SU[14];
input->m_su15=SU[15];
input->m_su16=SU[16];
input->m_su17=SU[17];
input->m_su18=SU[18];
input->m_su19=SU[19];
input->m_su20=SU[20];
input->m_total=TSU;
input->m_usesupp = SUPPL;
response=input->DoModal();
if (response==IDCANCEL){
delete input;
return;
}
GRCost[1]=input->m_grcost1;
GRCost[2]=input->m_grcost2;
GRCost[3]=input->m_grcost3;
GRCost[4] = input->m_grcost4;
GRCost[5] = input->m_grcost5;
GRCost[6] = input->m_grcost6;
GRCost[7] = input->m_grcost7;
GRCost[8] = input->m_grcost8;
GRCost[9] = input->m_grcost9;
GRCost[10] = input->m_grcost10;
GRCost[11] = input->m_grcost11;
GRCost[12] = input->m_grcost12;
GRCost[13] = input->m_grcost13;
GRCost[14] = input->m_grcost14;
GRCost[15] = input->m_grcost15;
GRCost[16] = input->m_grcost16;
GRCost[17] = input->m_grcost17;
GRCost[18] = input->m_grcost18;
GRCost[19] = input->m_grcost19;
GRCost[20] = input->m_grcost20;
SOPT1 = input->m_sopt1;
SOPT2 = input->m_sopt2;
SOPT3 = input->m_sopt3;
SOPT4 = input->m_sopt4;
SOPT5 = input->m_sopt5;
SU[1] = input->m_su1;
SU[2] = input->m_su2;
SU[3] = input->m_su3;
SU[4] = input->m_su4;
SU[5] = input->m_su5;
SU[6] = input->m_su6;
SU[7] = input->m_su7;
SU[8] = input->m_su8;
SU[9] = input->m_su9;
SU[10] = input->m_su10;
SU[11] = input->m_su11;
SU[12] = input->m_su12;
SU[13] = input->m_su13;
SU[14] = input->m_su14;
SU[15] = input->m_su15;
SU[16] = input->m_su16;
SU[17] = input->m_su17;
SU[18] = input->m_su18;
SU[19] = input->m_su19;
SU[20] = input->m_su20;
SUPPL = input->m_usesupp;
    SU[20];
if (TSU != 1)
    AfxMessageBox("Totals must equal 1", MB_OK);
delete input;
while (TSU != 1);
if (response==IDC_NEXT){
   UseInput8();
}

void Nutrition::UseInput8(){
CInput8* input=new CInput8;
int count;
input->m_sdm16=SDM[16];
input->m_sdm17=SDM[17];
input->m_sdm18=SDM[18];
input->m_sdm19=SDM[19];
input->m_sdm20=SDM[20];
input->m_sme16=SME[16];
input->m_sme17=SME[17];
input->m_sme18=SME[18];
input->m_sme19=SME[19];
input->m_sme20=SME[20];
input->m_sneg16=SNEG[16];
input->m_sneg17=SNEG[17];
input->m_sneg18=SNEG[18];
input->m_sneg19=SNEG[19];
input->m_sneg20=SNEG[20];
input->m_snem16=SNEM[16];
input->m_snem17=SNEM[17];
input->m_snem18=SNEM[18];
input->m_snem19=SNEM[19];
input->m_snem20=SNEM[20];
input->m_stdn16=STDN[16];
input->m_stdn17=STDN[17];
input->m_stdn18=STDN[18];
input->m_stdn19=STDN[19];
input->m_stdn20=STDN[20];
input->m_sopt1=SOPT1;
input->m_sopt2=SOPT2;
input->m_sopt3=SOPT3;
input->m_sopt4=SOPT4;
input->m_sopt5=SOPT5;
int response=input->DoModal();
if (response==IDCANCEL){
delete input;
return;
}
SDM[16]=input->m_sdm16;
SDM[17]=input->m_sdm17;
SDM[18]=input->m_sdm18;
SDM[19]=input->m_sdm19;
SDM[20]=input->m_sdm20;
SME[16]=input->m_sme16;
SME[17]=input->m_sme17;
SME[18]=input->m_sme18;
SME[19]=input->m_sme19;
SME[20]=input->m_sme20;
SNEG[16]=input->m_sneg16;
SNEG[17]=input->m_sneg17;
SNEG[18]=input->m_sneg18;
SNEG[19]=input->m_sneg19;
SNEG[20]=input->m_sneg20;
SNEM[16]=input->m_snem16;
SNEM[17]=input->m_snem17;
SNEM[18]=input->m_snem18;
SNEM[19]=input->m_snem19;
SNEM[20]=input->m_snem20;
STDN[16]=input->m_stdn16;
STDN[17]=input->m_stdn17;
STDN[18]=input->m_stdn18;
STDN[19]=input->m_stdn19;
STDN[20]=input->m_stdn20;
SOPT1=input->m_sopt1;
SOPT2=input->m_sopt2;
SOPT3=input->m_sopt3;
SOPT4=input->m_sopt4;
SOPT5=input->m_sopt5;
delete input;
for (count=0; count<21; count++){
    mainSDM += (SDM[count]*SU[count]);
    SAME += SME[count]*SU[count];
    SANEM += SNEM[count]*SU[count];
    SANEG += SNEG[count]*SU[count];
}
for (count=0; count<11; count++){
    mainSDMC += (SDM[count] * 
            +SU[8]+SU[9])));
    CSAME += (SME[count] * 
            +SU[8]+SU[9])));
    CSANEM += (SNEM[count] * 
            +SU[8]+SU[9])));
    CSANEG += (SNEG[count] * 
            +SU[8]+SU[9])));
}
if (response==IDC_PREV){
    UseInput7();
}
return;
void Nutrition::Init()
{
    UseInput7();
    cout<<"PRESS ENTER TO BEGIN THE SIMULATION"<<endl;
}

void Nutrition::FeedUsage()
{
    int count;
    if (currentDATE==1)
    {
        FeedCost=0;
        AdjFeedCost=0;
        STFeedCost=0;
        AdjSTFeedCost=0;
        hayRevenue=0;
    }
    if ((currentDATE ==365) && (YEAR>=1))
    {
        GREDCost=0;
        CGREDCost=0;
        for (count=0; count<21; count++)
        {
            GREDCost+=GRCost[count]*SU[count];
        }
        for (count=0; count<11; count++)
        {
            CGREDCost+=GRCost[count] * SU[count];
        }
        FeedCost = ( (((TPDMIS/mainSDM) * 2.20459) * GREDCost) +
        (((TCPDMIS/mainSDMC) * 2.20459) * CGREDCost) )
        STFeedCost = ( (TSPDMIS/mainSDM) * 2.20459) * GREDCost;
        hayRevenue=( (hayAVAIL/mainFDM) * (hayVALUE - hayCost) )
        AdjSTFeedCost = STFeedCost - hayRevenue;
        if (AdjSTFeedCost>=0)
        {
            STFeedCost=AdjSTFeedCost;
            FeedCost=FeedCost;
        }
        else if (AdjSTFeedCost<0)
        {
            AdjSTFeedCost=0;
            hayRevenue=hayRevenue-STFeedCost;
            AdjFeedCost=(FeedCost - hayRevenue );
            FeedCost=AdjFeedCost;
            STFeedCost=AdjSTFeedCost;
        }
    }
    if (currentDATE==365)
    {
        hayConump = ( (((TPDMIS/mainSDM) * 2.20459)* SU[0]) +
        (((TPDMIS/mainSDMC) * 2.20459)* SU[1])
        + (((TPDMIS/mainSDM) * 2.20459)* SU[2]) + (((TPDMIS/mainSDM)
        * 2.20459)* SU[3])
        + (((TPDMIS/mainSDM) * 2.20459)* SU[4]) + (((TPDMIS/mainSDM)
        * 2.20459)* SU[5])
        + (((TPDMIS/mainSDM) * 2.20459)* SU[6]) + (((TPDMIS/mainSDM)
        * 2.20459)* SU[7])
        + (((TPDMIS/mainSDM) * 2.20459)* SU[8]) + (((TPDMIS/mainSDM)
        * 2.20459)* SU[9])
        + (((TCPDMIS/mainSDMC) * 2.20459) *
+ (((TCPDMIS/mainSDMC) * 2.20459) *
+ (((TCPDMIS/mainSDMC) * 2.20459) *
+ (((TCPDMIS/mainSDMC) * 2.20459) *
+ (((TCPDMIS/mainSDMC) * 2.20459) *
+ (((TCPDMIS/mainSDMC) * 2.20459) *
))
Concentrate=( (((TPDMIS/mainSDM) * 2.20459)* SU[10])
+ (((TPDMIS/mainSDM) * 2.20459)* SU[11])
+ (((TPDMIS/mainSDM) * 2.20459)* SU[12])
+ (((TPDMIS/mainSDM) * 2.20459)* SU[13])
+ (((TPDMIS/mainSDM) * 2.20459)* SU[14])
+ (((TPDMIS/mainSDM) * 2.20459)* SU[15]));
hayConsumpS=( (((TSPDMIS/mainSDM) * 2.20459)* SU[0])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[1])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[2])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[3])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[4])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[5])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[6])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[7])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[8])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[9]));
ConcentrateS=( (((TSPDMIS/mainSDM) * 2.20459)* SU[10])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[11])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[12])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[13])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[14])
+ (((TSPDMIS/mainSDM) * 2.20459)* SU[15]));
void Nutrition::MainNutrition(AnimalRecord *animal){
    GUNIT=((CurrentSIZE+CurrentSIZES)/2.47);
    if (animal->SEX==0){
        if (IMPH==1){
            if (animal->REPLA==0){
                animal->IMPLANT=1;
            }
            else if (animal->REPLA==1){
                animal->IMPLANT=0;
            }
        }
        else if (IMPH==0){
            animal->IMPLANT=0;
        }
        else if (animal->SEX==2){
            if(IMPS==1){
                animal->IMPLANT=1;
            }
            else if (IMPS==0){
                animal->IMPLANT=0;
            }
        }
        else if (animal->SEX==3){
            animal->IMPLANT==0;
        }
        else if (animal->SEX==4){
            if (IMPHS==1){
                if (animal->REPLA==0){
                    animal->IMPLANT=1;
                }
                else if (animal->REPLA==1){
                    animal->IMPLANT=0;
                }
            }
            else if (IMPHS==0){
                animal->IMPLANT=0;
            }
        }
        else if (animal->SEX==5){
            if (IMPSS==1){
                animal->IMPLANT=1;
            }
            else if (IMPSS==0){
                animal->IMPLANT=0;
            }
        }
    }
}
if (currentDATE==1){
APDMIF=0;
ASPDMIF=0;
TPDMIS=0;
TSPDMIS=0;
}
if (PAVAIL>0){
if ((animal->AGE <= 365) && (animal->SEX <= 3)){
PDMIF = ((pow(animal->SBWTK,.75 )) * (((.2435 * (FANEM)) - (.0466*(pow(FANEM,2))) -.1128)/FANEM));
}
else if ((animal->AGE<365) && (animal->SEX > 3)){
SPDMIF = ((pow(animal->SBWTK,.75 )) * (((.2435 * (FANEM)) - (.0466*(pow(FANEM,2))) -.1128)/FANEM));
}
else if ((animal->AGE >=365) && (animal->SEX==2)){
PDMIF = ((pow(animal->SBWTK,.75 )) * (((.2435 * (FANEM)) - (.0466*(pow(FANEM,2))) -.0869)/FANEM));
}
else if ((animal->AGE >=365) && (animal->SEX==5)){
SPDMIF = ((pow(animal->SBWTK,.75 )) * (((.2435 * (FANEM)) - (.0466*(pow(FANEM,2))) -.0869)/FANEM));
}
else if ((animal->AGE >=365) && (animal->AGE < 913) && (animal->SEX==0)){
PDMIF = ((pow(animal->SBWTK,.75 )) * (((.04997 * (pow(FANEM,.2)))+.0384)/FANEM));
}
else if ((animal->AGE >=365) && (animal->AGE < 913) && (animal->SEX==4)){
SPDMIF = ((pow(animal->SBWTK,.75 )) * (((.04997 * (pow(FANEM,.2)))+.0384)/FANEM));
}
else if ( (animal->AGE>=365) && (animal->AGE < 913) && (animal->SEX==3) && (animal->PREG==0) && (FANEM >= 1) && (animal->LACT==0)){
PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM,.2)))+.0384)/FANEM));
}
else if ( (animal->AGE>=365) && (animal->AGE < 913) && (animal->SEX==3) && (animal->PREG==0) && (FANEM < 1) && (animal->LACT==0)){
PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(.95,.2)))+.0384)/.95));
}
else if ( (animal->AGE>=365) && (animal->AGE < 913) && (animal->SEX==3) && (animal->PREG==1) && (FANEM >= 1) && (animal->DPREG <= 95 && (animal->LACT==0)) ){ 
PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM,.2)))+.0384)/FANEM));
}
else if ((animal->AGE >= 365) &&
          (animal->AGE < 913) &&
          (animal->SEX == 3) &&
          (animal->PREG == 1) &&
          (animal->DPREG == 0) && (animal->LACT == 0)){
      PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(.95, 2))) + .0384) / .95));
    }
else if ((animal->AGE >= 365) &&
          (animal->AGE < 913) &&
          (animal->SEX == 3) &&
          (animal->PREG == 1) &&
          (animal->DPREG > 95) && (animal->LACT == 0)){
      PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .04361) / FANEM));
    }
else if ((animal->AGE >= 365) &&
          (animal->AGE <= 913) &&
          (animal->SEX == 3) &&
          (animal->PREG == 1) &&
          (animal->DPREG > 95) && (animal->LACT == 1)){
      PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(.95, 2))) + .04361) / .95));
    }
else if ((animal->AGE >= 365) &&
          (animal->AGE <= 913) &&
          (animal->SEX == 3) &&
          (animal->PREG == 0) &&
          (animal->LACT == 1) &&
          (FANEM >= 1)){
      PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .04361) / FANEM));
    }
else if ((animal->AGE >= 365) &&
          (animal->AGE <= 913) &&
          (animal->SEX == 3) &&
          (animal->PREG == 0) &&
          (animal->LACT == 1) &&
          (FANEM < 1)){
      PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(.95, 2))) + .04361) / .95));
    }
else if ((animal->AGE == 913) &&
          (animal->SEX == 3) &&
          (animal->PREG == 0) &&
          (FANEM >= 1) &&
          (animal->LACT == 0)){
      PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .0384) / FANEM));
    }
else if ((animal->AGE == 913) &&
          (animal->SEX == 3) &&
          (animal->PREG == 0) &&
          (FANEM < 1) &&
          (animal->LACT == 0)){
      PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(.95, 2))) + .0384) / .95));
    }

if ((animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (animal->LACT==0))
    PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .0384) / FANEM));
}
else if ((animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (FANEM < 1) && (animal->DPREG <= 95) && (animal->LACT==0))
    PDMIF = (pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .0384) / FANEM));
else if ((animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (animal->LACT==0))
    PDMIF = (pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .04361) / FANEM));
else if ( (animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (animal->LACT==1) && (FANEM >= 1))
    PDMIF = (pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .04361) / FANEM));
else if ( (animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (animal->LACT==1) && (FANEM < 1))
    PDMIF = (pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .04361) / FANEM));
else if ( (animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==0) && (animal->LACT==1) && (FANEM >= 1))
    PDMIF = (pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .04361) / FANEM));
else if ( (animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==0) && (FANEM < 1))
    PDMIF = ((pow(animal->SBWTK, .75)) * (((.04997 * (pow(FANEM, 2))) + .04361) / FANEM));
else if ((animal->IMPLANT==0) && (animal->SEX < 3))
    PDMIF= (PDMIF * .94);
else if ((animal->IMPLANT==0) && (animal->SEX > 3))

SPDMIF=(SPDMIF * .94);
}
if ((animal->EQEBW <= 350) && (animal->AGE < 548) && (animal->SEX <= 3)){
PDMIF=PDMIF * 1;
}
else if ((animal->EQEBW > 350) && (animal->EQEBW <=400) && (animal->AGE < 548) && (animal->SEX <= 3)){
PDMIF=PDMIF * .97;
}
else if ((animal->EQEBW > 400) && (animal->EQEBW <=450) && (animal->AGE < 548) && (animal->SEX <= 3)){
PDMIF=PDMIF * .9;
}
else if ((animal->EQEBW > 450) && (animal->EQEBW <=500) && (animal->AGE < 548) && (animal->SEX <= 3)){
PDMIF=PDMIF * .82;
}
else if ((animal->EQEBW > 500) && (animal->EQEBW <=550) && (animal->AGE < 548) && (animal->SEX <= 3)){
PDMIF=PDMIF * .73;
}
else if ((animal->EQEBW <= 350) && (animal->AGE >= 548) && (animal->SEX < 3)){
SPDMIF=SPDMIF * 1;
}
else if ((animal->EQEBW > 350) && (animal->EQEBW <=400) && (animal->AGE >= 548) && (animal->SEX < 3)){
PDMIF=PDMIF * .97;
}
else if ((animal->EQEBW > 400) && (animal->EQEBW <=450) && (animal->AGE >= 548) && (animal->SEX < 3)){
PDMIF=PDMIF * .9;
}
else if ((animal->EQEBW > 450) && (animal->EQEBW <=500) && (animal->AGE >= 548) && (animal->SEX < 3)){
PDMIF=PDMIF * .82;
}
else if ((animal->EQEBW > 500) && (animal->EQEBW <=550) && (animal->AGE >= 548) && (animal->SEX < 3)){
PDMIF=PDMIF * .73;
}
else if ((animal->EQEBW <= 350) && (animal->AGE < 548) && (animal->SEX > 3)){
SPDMIF=SPDMIF * 1;
}
else if ((animal->EQEBW > 350) && (animal->EQEBW <=400) && (animal->AGE < 548) && (animal->SEX > 3)){
SPDMIF=SPDMIF * .97;
}
else if ((animal->EQEBW > 400) && (animal->EQEBW <= 450) &&

(animal->AGE < 548) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * .9;
} else if ((animal->EQEBW > 450) && (animal->EQEBW <= 500) && 
    (animal->AGE < 548) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * .82;
} else if ((animal->EQEBW > 500) && (animal->EQEBW <= 550) && 
    (animal->AGE < 548) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * .73;
} else if ((animal->EQEBW <= 350) && (animal->AGE >= 548) && 
    (animal->SEX > 3)) {
    SPDMIF = SPDMIF * 1;
} else if ((animal->EQEBW > 350) && (animal->EQEBW <= 400) && 
    (animal->AGE >= 548) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * .97;
} else if ((animal->EQEBW > 400) && (animal->EQEBW <= 450) && 
    (animal->AGE >= 548) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * .9;
} else if ((animal->EQEBW > 450) && (animal->EQEBW <= 500) && 
    (animal->AGE >= 548) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * .82;
} else if ((animal->EQEBW > 500) && (animal->EQEBW <= 550) && 
    (animal->AGE >= 548) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * .73;
} 

if ((Tc >= 25) && (animal->SEX <= 3)) {
    PDMIF = PDMIF * .9;
} else if ((Tc >= 15) && (Tc < 25) && (animal->SEX <= 3)) {
    PDMIF = PDMIF * 1;
} else if ((Tc >= 5) && (Tc < 15) && (animal->SEX <= 3)) {
    PDMIF = PDMIF * 1.03;
} else if ((Tc >= -5) && (Tc < 5) && (animal->SEX <= 3)) {
    PDMIF = PDMIF * 1.05;
} else if ((Tc >= -15) && (Tc < -5) && (animal->SEX <= 3)) {
    PDMIF = PDMIF * 1.16;
} else if ((Tc >= 25) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * .9;
} else if ((Tc >= 15) && (Tc < 25) && (animal->SEX > 3)) {
    SPDMIF = SPDMIF * 1;
SPDMIF = SPDMIF * 1;
} else if ((Tc >= 5) && (Tc < 15) && (animal->SEX > 3)){
SPDMIF = SPDMIF * 1.03;
} else if ((Tc >= -5) && (Tc < 5) && (animal->SEX > 3)){
SPDMIF = SPDMIF * 1.05;
} else if ((Tc >= -15) && (Tc < -5) && (animal->SEX > 3)){
SPDMIF = SPDMIF * 1.16;
} if ( (animal->SEX==3) && (animal->PREG==1) && (animal->LACT==1) ){
PDMIF = (PDMIF + (.2 * (animal->YN)) );
} if ( (PDMIF>PAVAIL) && (PAVAIL>=0) && (animal->SEX<=3) ){
PDMIF=PAVAIL;
PAVAIL-=PDMIF;
TPDMIF+=PDMIF;
} else if ( (SPDMIF>PAVAIL) && (PAVAIL>=0) && (animal->SEX>3) ){
SPDMIF=PAVAIL;
PAVAIL-=SPDMIF;
TSPDMIF+=SPDMIF;
} else if ( (PAVAIL>=PDMIF) && (animal->SEX<=3) ){
PAVAIL-=PDMIF;
TPDMIF+=PDMIF;
} else if ( (PAVAIL>=SPDMIF) && (animal->SEX>3) ){
PAVAIL-=SPDMIF;
TSPDMIF+=SPDMIF;
} if (PAVAIL>0){
DIPM=(PAVAIL/2.47);
} else{
DIPM=0;
} if (DIPM>0){
if (((DIPM/GUNIT) < 1150) && ((DIPM/GUNIT) > 0) && (SUPPL==1)){
GRAZE=(((.17*(DIPM/GUNIT))-.000074*pow((DIPM/GUNIT), 2))+2.4)/100);
SUPP=1-GRAZE;
if (((DIPM/COWS) < 50) && ((DIPM/COWS) > 0)){
SUPP  = 1 * SUPP;
GRAZE = 1 - SUPP;
} else if (((DIPM/COWS) >= 50) && ((DIPM/COWS) < 100)){

902
SUPP = .8 * SUPP;
GRAZE = 1 - SUPP;
}
else if (((DIPM/COWS) >= 100) && ((DIPM/COWS) < 150)) {
    SUPP = .6 * SUPP;
    GRAZE = 1 - SUPP;
}
else if (((DIPM/COWS) >= 150) && ((DIPM/COWS) < 200)) {
    SUPP = .4 * SUPP;
    GRAZE = 1 - SUPP;
}
else if (((DIPM/COWS) >= 200) && ((DIPM/COWS) < 250)) {
    SUPP = .2 * SUPP;
    GRAZE = 1 - SUPP;
}
else if ((DIPM/COWS) >= 250) {
    SUPP = 0;
    GRAZE = 1;
}
}
else if (((DIPM/GUNIT) >= 1150) && (SUPPL==1)) {
    GRAZE=1;
    SUPP=0;
}
else if (((DIPM/GUNIT) < 1150) || ((DIPM/GUNIT) >= 1150) && (SUPPL==0)) {
    GRAZE=1;
    SUPP=0;
}
else if (DIPM<=0) {
    SUPP=1;
    GRAZE=0;
}
if ((animal->SEX <= 3) && (IONO==0)) {
    PDMIFA = GRAZE * PDMIF;
    PAVAIL += (PDMIF - PDMIFA);
    APDMIF += PDMIFA;
    TPDMIFA=APDMIF;
}
else if ((animal->SEX > 3) && (IONO==0)) {
    SPDMIFA = GRAZE * SPDMIF;
    PAVAIL += (SPDMIF - SPDMIFA);
    ASPDMIF += SPDMIFA;
    TSPDMIFA=ASPDMIF;
}
else if ((animal->SEX <= 3) && (IONO==1)) {
    PDMIFA = GRAZE * PDMIF * (1/1.12);
    PAVAIL += (PDMIF - PDMIFA);
    APDMIF += PDMIFA;
    TPDMIFA=APDMIF;
else if ((animal->SEX > 3) && (IONO==1)){
    SPDMIFA = GRAZE * SPDMIF * (1/1.12);
    PAVAIL += (SPDMIF - SPDMIFA);
    ASPDMIF += SPDMIFA;
    TSPDMIFA=ASPDMIFA;
}

if ( (GRAZE < 1) && (GRAZE>=0) && (SUPPL == 1)){
    if ((animal->AGE <= 365) && (animal->SEX < 3)){
        PDMIS = (((pow(animal->SBWTK,.75 )) * (((.2435 * (SANEM)) - (.0466*(pow(SANEM,2))) -.1128)/SANEM)) * SUPP);
    }
    else if ((animal->AGE < 365) && (animal->SEX > 3)){
        SPDMIS = (((pow(animal->SBWTK,.75 )) * (((.2435 * (SANEM)) - (.0466*(pow(SANEM,2))) -.1128)/SANEM)) * SUPP);
    }
    else if ((animal->AGE >=365) && (animal->SEX < 3)){
        PDMIS = (((pow(animal->SBWTK,.75 )) * (((.2435 * (SANEM)) - (.0466*(pow(SANEM,2))) -.0869)/SANEM)) * SUPP);
    }
    else if ((animal->AGE >=365) && (animal->SEX > 3)){
        SPDMIS = (((pow(animal->SBWTK,.75 )) * (((.2435 * (SANEM)) - (.0466*(pow(SANEM,2))) -.0869)/SANEM)) * SUPP);
    }
    else if ((animal->AGE <= 365) && (animal->SEX == 3)){
        CPDMIS = (((pow(animal->SBWTK, .75)) * (((.04997 *(pow(CSANEM, 2)))+.0384)/CSANEM))*SUPP);
    }
    else if ((animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==0) && (CSANEM < 1)){
        CPDMIS = (((pow(animal->SBWTK, .75)) * (((.04997 * (pow(.95, 2)))+.0384)/.95))*SUPP);
    }
    else if ((animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (CSANEM >= 1) && (animal->DPREG <= 95)){
        CPDMIS = (((pow(animal->SBWTK, .75)) * (((.04997 * (pow(CSANEM, 2)))+.0384)/CSANEM))*SUPP);
    }
    else if ((animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (CSANEM < 1) && (animal->DPREG <= 95)){
        CPDMIS = (((pow(animal->SBWTK, .75)) * (((.04997 * (pow(.95, 2)))+.0384)/.95))*SUPP);
    }
    else if ((animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (CSANEM == 0) && (animal->DPREG <= 95)){
        CPDMIS = (((pow(animal->SBWTK, .75)) * (((.04997 * (pow(CSANEM, 2)))+.0384)/CSANEM))*SUPP);
    }
}
else if ((animal->AGE >= 913) && (animal->SEX==3) && (animal->PREG==1) && (CSANEM < 1) && (animal->DPREG > 95)){
    CPDMIS = (((pow(animal->SBWTK, .75)) * (((.04997 * (pow(.95, 2)))+.04361)/.95))*SUPP);
}

if ((animal->IMPLANT==0) && (animal->SEX < 3)){
    PDMIS=(PDMIS * .94);
}

if ((animal->IMPLANT==0) && (animal->SEX > 3)){
    SPDMIS=(SPDMIS * .94);
}

if ((animal->EQEBW <= 350) && (animal->AGE < 548) && (animal->SEX < 3)){
    PDMIS=PDMIS * 1;
}

else if ((animal->EQEBW > 350) && (animal->EQEBW <=400) && (animal->AGE < 548) && (animal->SEX < 3)){
    PDMIS=PDMIS * .97;
}

else if ((animal->EQEBW > 400) && (animal->EQEBW <=450) && (animal->AGE < 548) && (animal->SEX < 3)){
    PDMIS=PDMIS * .9;
}

else if ((animal->EQEBW > 450) && (animal->EQEBW <=500) && (animal->AGE < 548) && (animal->SEX < 3)){
    PDMIS=PDMIS * .82;
}

else if ((animal->EQEBW > 500) && (animal->EQEBW <=550) && (animal->AGE < 548) && (animal->SEX < 3)){
    PDMIS=PDMIS * .73;
}

if ((animal->EQEBW <= 350) && (animal->AGE < 548) && (animal->SEX == 3)){
    CPDMIS=CPDMIS * 1;
}

else if ((animal->EQEBW > 350) && (animal->EQEBW <=400) && (animal->AGE < 548) && (animal->SEX == 3)){
    CPDMIS=CPDMIS * .97;
}

else if ((animal->EQEBW > 400) && (animal->EQEBW <=450) && (animal->AGE < 548) && (animal->SEX == 3)){
    CPDMIS=CPDMIS * .9;
}

else if ((animal->EQEBW > 450) && (animal->EQEBW <=500) && (animal->AGE < 548) && (animal->SEX == 3)){

(animal->AGE < 548) && (animal->SEX == 3)) {
  CPDMIS = CPDMIS * .82;
}
else if (((animal->EQEBW > 500) && (animal->EQEBW <= 550) &&
            (animal->AGE < 548) && (animal->SEX == 3)) {
  CPDMIS = CPDMIS * .73;
}
else if (((animal->EQEBW <= 350) && (animal->AGE >= 548) &&
            (animal->SEX < 3)) {
  PDMIS = PDMIS * 1;
}
else if (((animal->EQEBW > 350) && (animal->EQEBW <= 400) &&
            (animal->AGE >= 548) && (animal->SEX < 3)) {
  PDMIS = PDMIS * .97;
}
else if (((animal->EQEBW > 400) && (animal->EQEBW <= 450) &&
            (animal->AGE >= 548) && (animal->SEX < 3)) {
  PDMIS = PDMIS * .9;
}
else if (((animal->EQEBW > 450) && (animal->EQEBW <= 500) &&
            (animal->AGE >= 548) && (animal->SEX < 3)) {
  PDMIS = PDMIS * .82;
}
else if (((animal->EQEBW > 500) && (animal->EQEBW <= 550) &&
            (animal->AGE >= 548) && (animal->SEX < 3)) {
  PDMIS = PDMIS * .73;
}
else if (((animal->EQEBW <= 350) && (animal->AGE < 548) &&
            (animal->SEX > 3)) {
  SPDMIS = SPDMIS * 1;
}
else if (((animal->EQEBW > 350) && (animal->EQEBW <= 400) &&
            (animal->AGE < 548) && (animal->SEX > 3)) {
  SPDMIS = SPDMIS * .97;
}
else if (((animal->EQEBW > 400) && (animal->EQEBW <= 450) &&
            (animal->AGE < 548) && (animal->SEX > 3)) {
  SPDMIS = SPDMIS * .9;
}
else if (((animal->EQEBW > 450) && (animal->EQEBW <= 500) &&
            (animal->AGE < 548) && (animal->SEX > 3)) {
  SPDMIS = SPDMIS * .82;
}
else if (((animal->EQEBW > 500) && (animal->EQEBW <= 550) &&
            (animal->AGE < 548) && (animal->SEX > 3)) {
  SPDMIS = SPDMIS * .73;
}
else if (((animal->EQEBW <= 350) && (animal->AGE >= 548) &&
            (animal->SEX > 3)) {
  SPDMIS = SPDMIS * 1;
}
else if ((animal->EQEBW > 350) && (animal->EQEBW <= 400) &&
(animal->AGE >= 548) && (animal->SEX > 3)){
SPDMIS = SPDMIS * .97;
}
else if ((animal->EQEBW > 400) && (animal->EQEBW <= 450) &&
(animal->AGE >= 548) && (animal->SEX > 3)){
SPDMIS = SPDMIS * .9;
}
else if ((animal->EQEBW > 450) && (animal->EQEBW <= 500) &&
(animal->AGE >= 548) && (animal->SEX > 3)){
SPDMIS = SPDMIS * .82;
}
else if ((animal->EQEBW > 500) && (animal->EQEBW <= 550) &&
(animal->AGE >= 548) && (animal->SEX > 3)){
SPDMIS = SPDMIS * .73;
}

if ((Tc >= 25) && (animal->SEX < 3)){
PDMIS = PDMIS * .9;
}
else if ((Tc >= 15) && (Tc < 25) && (animal->SEX < 3)){
PDMIS = PDMIS * 1;
}
else if ((Tc >= 5) && (Tc < 15) && (animal->SEX < 3)){
PDMIS = PDMIS * 1.03;
}
else if ((Tc >= -5) && (Tc < 5) && (animal->SEX < 3)){
PDMIS = PDMIS * 1.05;
}
else if ((Tc >= -15) && (Tc < -5) && (animal->SEX < 3)){
PDMIS = PDMIS * 1.16;
}
else if ((Tc >= 25) && (animal->SEX == 3)){
CPDMIS = CPDMIS * .9;
}
else if ((Tc >= 15) && (Tc < 25) && (animal->SEX == 3)){
CPDMIS = CPDMIS * 1;
}
else if ((Tc >= 5) && (Tc < 15) && (animal->SEX == 3)){
CPDMIS = CPDMIS * 1.03;
}
else if ((Tc >= -5) && (Tc < 5) && (animal->SEX == 3)){
CPDMIS = CPDMIS * 1.05;
}
else if ((Tc >= -15) && (Tc < -5) && (animal->SEX == 3)){
CPDMIS = CPDMIS * 1.16;
}
else if ((Tc >= 25) && (animal->SEX > 3)){
SPDMIS = SPDMIS * .9;
}
else if ((Tc >= 15) && (Tc < 25) && (animal->SEX > 3)){
SPDMIS = SPDMIS * 1;
else if ((Tc >= 5) && (Tc < 15) && (animal->SEX > 3)){
    SPDMIS = SPDMIS * 1.03;
}
else if ((Tc >= -5) && (Tc < 5) && (animal->SEX > 3)){
    SPDMIS = SPDMIS * 1.05;
}
else if ((Tc >= -15) && (Tc < -5) && (animal->SEX > 3)){
    SPDMIS = SPDMIS * 1.16;
}
else if ((animal->SEX==3) && (animal->PREG==1) && (animal->LACT==1)){
    CPDMIS = (CPDMIS + (.2 * (animal->YN)));
}
else if ( (animal->SEX<3) && (PDMIS>=0) ){
    TPDMIS += PDMIS;
}
else if ( (animal->SEX == 3) && (CPDMIS>=0) ){
    TCPDMIS += CPDMIS;
}
else if ( (animal->SEX > 3) && (SPDMIS>=0) ){
    TSPDMIS += SPDMIS;
}

void Nutrition::DWGAIN(AnimalRecord *animal){
    double A1 = 0.077;
    double A2=.0007;
    double PKW[8]={6,7,8,8.5,9,11,11.5,12};
    double AL=0;
    double CNEM=0;
    double CNEMAP=0;
    double CNEMP=0;
    double CDMIEQ=0;
    double E=0;
    double FatCompFac=0;
    double GUWTKD=0;
    double IM=0;
    double KL=0;
    double ME=0;
    double NEM=0;
    double NEG=0;
    double PDMI=0;
    double CPDMI=0;
    double SPDMI=0;
    double RE=0;
    double SBWTG=0;
}
double KM=0;
double InternInsul=0;
double ExternInsul=0;
double Insulate=0;
double SurArea=0;
double MEIntake=0;
double HeatProd=0;
double LowCritTemp=0;
double CNEMCS=0;
int WIND = 10;
double HAIR=0;
int HIDE=0;
double excessSBWTG=0;
double extraRE=0;
double excessSPDMI=0;
double r=0;
double tempSBWTK=animal->SBWTK;
double tempCDWTK=animal->CDWTK;
double tempEQSBW=animal->EQSBW;
NEM = ((FANEM * GRAZE) + (SANEM * SUPP));
NEG = ((FANEG * GRAZE) + (SANEG * SUPP));
ME = ((FAME * GRAZE) + (SAME * SUPP));
PDMI = (PDMIFA + PDMIS);
CPDMI = (PDMIFA + CPDMIS);
SPDMI = (SPDMIFA + SPDMIS);
CDMIEQ = ( ( ((double) animal->YEN) + (PDMI * NEM) ) / NEM);
if ( (animal->BREED == 2) || (animal->BREED==3) ){
HIDE=3;
}
else{
HIDE= 2;
}
if ( (currentDATE>= 1) && (currentDATE< 121) ){
HAIR = .5 * .45359;
}
else if ( (currentDATE>=121) && (currentDATE < 274) ){
HAIR = .2 * .45359;
}
else if ( (currentDATE >= 274) && (currentDATE <=365) ){
HAIR = .5 * .45359;
}
if ( ( animal->SEX==3) && (animal->LACT==1) && (animal->LDAY==1) ) {
animal->MCMFC = GetStochasticNumber(1.24, 4.03);
animal->MCSNF = GetStochasticNumber (1.38, 8.31);
}
if ( (animal->SEX==3)  && ( (animal->CDWT>animal->PMWT) ||
( (animal->SBWTK - animal->GUWTK) <= (((animal->PMWT*.4535)*.96) * .79)) ){
FatCompFac= .80 + ((1-1) * .05);
animal->BCS=1;
}
else if (((animal->SBWTK - animal->GUWTK) >
((animal->PMWT*.4535)*.96) * .79)) &&
((animal->SBWTK - animal->GUWTK) <=
((animal->PMWT*.4535)*.96) * .83)) {
  FatCompFac = .80 + ((2-1) * .05);
  animal->BCS=2;
}
else if (((animal->SBWTK - animal->GUWTK) >
((animal->PMWT*.4535)*.96) * .83)) &&
((animal->SBWTK - animal->GUWTK) <=
((animal->PMWT*.4535)*.96) * .90)) {
  FatCompFac = .80 + ((3-1) * .05);
  animal->BCS=3;
}
else if (((animal->SBWTK - animal->GUWTK) >
((animal->PMWT*.4535)*.96) * .90)) &&
((animal->SBWTK - animal->GUWTK) <=
((animal->PMWT*.4535)*.96) * .96)) {
  FatCompFac = .80 + ((4-1) * .05);
  animal->BCS=4;
}
else if (((animal->SBWTK - animal->GUWTK) >
((animal->PMWT*.4535)*.96) * .96)) &&
((animal->SBWTK - animal->GUWTK) <=
((animal->PMWT*.4535)*.96) * 1.04)) {
  FatCompFac = .80 + ((5-1) * .05);
  animal->BCS=5;
}
else if (((animal->SBWTK - animal->GUWTK) >
((animal->PMWT*.4535)*.96) * 1.04)) &&
((animal->SBWTK - animal->GUWTK) <=
((animal->PMWT*.4535)*.96) * 1.13)) {
  FatCompFac = .80 + ((6-1) * .05);
  animal->BCS=6;
}
else if (((animal->SBWTK - animal->GUWTK) >
((animal->PMWT*.4535)*.96) * 1.13)) &&
((animal->SBWTK - animal->GUWTK) <=
((animal->PMWT*.4535)*.96) * 1.24)) {
  FatCompFac = .80 + ((7-1) * .05);
  animal->BCS=7;
}
else if (((animal->SBWTK - animal->GUWTK) >
((animal->PMWT*.4535)*.96) * 1.24)) &&
((animal->SBWTK - animal->GUWTK) <=
((animal->PMWT*.4535)*.96) * 1.37)) {
  FatCompFac = .80 + ((8-1) * .05);
  animal->BCS=8;
}
else if ( (animal->SBWTK - animal->GUWTK) >
((animal->PMWT*.4535)*.96) * 1.37)) {
  FatCompFac = .80 + ((9-1) * .05);
  animal->BCS=9;
}
}
if (animal->SEX != 3){
if (animal->BREED == 11) {
    CNEM = (((A1 * (pow(tempSBWTK, .75))) * 1.2 * FatCompFac) + (A2 * (20-Tp)));
} 
else if (animal->BREED != 11) {
    CNEM = (((A1 * (pow(tempSBWTK, .75))) * 1 * FatCompFac) + (A2 * (20-Tp)));
}
else if ((animal->SEX == 3) && (animal->LACT == 0)) {
    if (animal->BREED == 11) {
        CNEM = (((A1 * (pow(tempSBWTK, .75))) * 1.2 * FatCompFac) + (A2 * (20-Tp)));
    } 
    else if (animal->BREED != 11) {
        CNEM = (((A1 * (pow(tempSBWTK, .75))) * 1 * FatCompFac) + (A2 * (20-Tp)));
    }
}
else if ((animal->SEX == 3) && (animal->LACT == 1)) {
    if (animal->BREED == 11) {
        CNEM = (((A1 * (pow(tempSBWTK, .75))) * 1.2 * 1.2 * FatCompFac) + (A2 * (20-Tp)));
    } 
    else if (animal->BREED != 11) {
        CNEM = (((A1 * (pow(tempSBWTK, .75))) * 1.2 * FatCompFac) + (A2 * (20-Tp)));
    }
}
if ((animal->PAST == 1) && (animal->SEX <= 3)) {
    CNEMAP = (((.006 * PDMIFA * (.9 - TDNP)) + (.05*TERR)/((DIPM/1000)+3)) * (animal->CDWTK/4.184));
}
else if ((animal->PAST == 1) && (animal->SEX > 3)) {
    CNEMAP = (((.006 * SPDMIFA * (.9 - TDNP)) + (.05*TERR)/((DIPM/1000)+3)) * (animal->CDWTK/4.184));
}
else if (animal->PAST == 0) {
    CNEMAP = 0;
}
if ((animal->LACT == 1) && (animal->SEX == 3)) {
    if (animal->BREED == 1) {
        KL=(1/PKW[1]);
        AL=(1/(8*KL*2.7183));
        animal->YN=((animal->LDAY/7) / (AL*(pow(2.7183,(KL * (animal->LDAY / 7))))));
        E=(((((.092 * animal->MCMFC) + (.049 * animal->MCSNF)) - .0569)) * 1);
    } 
    else if (animal->BREED == 2) {
        KL=(1/PKW[2]);
    }
AL = (1/(7*KL*2.7183));
animal->YN = ((animal->LDAY/7) / (AL*(pow(2.7183,(KL*(animal->LDAY)/7)))))
E = (((0.092 * animal->MCMFC) + (0.049 * animal->MCSNF)) - (0.0569)*1);
}
else if (animal->BREED == 3){
  KL = (1/PKW[3]);
  AL = (1/(7*KL*2.7183));
animal->YN = ((animal->LDAY/7) / (AL*(pow(2.7183,(KL*(animal->LDAY)/7)))))
E = (((0.092 * animal->MCMFC) + (0.049 * animal->MCSNF)) - (0.0569)*1);
}
else if (animal->BREED == 4){
  KL = (1/PKW[4]);
  AL = (1/(6.5*KL*2.7183));
animal->YN = ((animal->LDAY/7) / (AL*(pow(2.7183,(KL*(animal->LDAY)/7)))))
E = (((0.092 * animal->MCMFC) + (0.049 * animal->MCSNF)) - (0.0569)*1);
E = (E *.94);
}
else if (animal->BREED == 5){
  KL = (1/PKW[5]);
  AL = (1/(9*KL*2.7183));
animal->YN = ((animal->LDAY/7) / (AL*(pow(2.7183,(KL*(animal->LDAY)/7)))))
E = (((0.092 * animal->MCMFC) + (0.049 * animal->MCSNF)) - (0.0569)*.94);
E = (E *.94);
}
else if (animal->BREED == 6){
  KL = (1/PKW[6]);
  AL = (1/(6*KL*2.7183));
animal->YN = ((animal->LDAY/7) / (AL*(pow(2.7183,(KL*(animal->LDAY)/7)))))
E = (((0.092 * animal->MCMFC) + (0.049 * animal->MCSNF)) - (0.0569)*.91);
}
else if (animal->BREED == 7){
  KL = (1/(11.5*KL*2.7183));
animal->YN = ((animal->LDAY/7) / (AL*(pow(2.7183,(KL*(animal->LDAY)/7)))))
E = (((0.092 * animal->MCMFC) + (0.049 * animal->MCSNF)) - (0.0569)*1.27);
}
else if (animal->BREED == 8){
  KL = (1/PKW[8]);
  AL = (1/(9*KL*2.7183));
animal->YN = ((animal->LDAY/7) / (AL*(pow(2.7183,(KL*
E=(((0.092 * (animal->MCMFC) + 0.049 * (animal->MCSNF)) - 0.0569) * 0.91);
}
else if (animal->BREED==9){
KL=(1/PKW[9]);
AL=(1/((9*KL*2.7183)));
animal->YN=((animal->LDAY/7) / (AL*(pow( 2.7183, (KL * (animal->LDAY) / 7))))) );
E=(((0.092 * (animal->MCMFC) + 0.049 * (animal->MCSNF)) - 0.0569) * 1);
}
else if (animal->BREED==10){
KL=(1/PKW[10]);
AL=(1/((11*KL*2.7183)));
animal->YN=((animal->LDAY/7) / (AL*(pow( 2.7183, (KL * (animal->LDAY) / 7))))) );
E=(((0.092 * (animal->MCMFC) + 0.049 * (animal->MCSNF)) - 0.0569) * 1.11);
}
else if (animal->BREED==11){
KL=(1/PKW[11]);
AL=(1/((12*KL*2.7183)));
animal->YN=((animal->LDAY/7) / (AL*(pow( 2.7183, (KL * (animal->LDAY) / 7))))) );
E=(((0.092 * (animal->MCMFC) + 0.049 * (animal->MCSNF)) - 0.0569) * 1.33);
}
else if (animal->BREED==12){
KL=(1/PKW[12]);
AL=(1/((9*KL*2.7183)));
animal->YN=((animal->LDAY/7) / (AL*(pow( 2.7183, (KL * (animal->LDAY) / 7))))) );
E=(((0.092 * (animal->MCMFC) + 0.049 * (animal->MCSNF)) - 0.0569));
}
if ((animal->AGE >= 730) && (animal->AGE < 1095)) {
animal->YN = (animal->YN) * .74 ;
animal->YEN = (E * (animal->YN));
}
else if ((animal->AGE >= 1095) && (animal->AGE < 1460)) {
animal->YN = (animal->YN) * .88;
animal->YEN = (E * (animal->YN));
}
else if (animal->AGE >= 1460){
animal->YN = (animal->YN) * 1.0;
animal->YEN = (E * (animal->YN));
}
else if (animal->LACT!==1){
animal->YN=0;
}
animal->YEN=0;
}
if ( (animal->PREG==1) && (animal->SEX==3)) {
    if (animal->DPREG==0) {
        CNEMP=0;
        animal->GUWTK=0;
    } else if (animal->DPREG!=0) {
        CNEMP=(((animal->CBWT * .453599) * ((NEM/ME)/.13))*(0.05855 -
            (0.0000996 * animal->DPREG)) * (pow(2.7183 , (0.03233 - (0.000275 *
            animal->DPREG)) / 1000 ));
        GUWTKD=(((animal->CBWT) * .453599) * (.3656 - (0.000523 *
            animal->DPREG)) * (pow(2.71881828,( (.02 * (animal->DPREG)) - (0.0000143 * (pow((animal->DPREG),2)))))))/1000  );
        animal->GUWTK+=GUWTKD;
    }
} else if ( (animal->PREG == 0) && (animal->SEX == 3)) {
    CNEMP=0;
    GUWTKD=0;
    animal->GUWTK=0;
} if ((IONS==0) && (animal->SEX < 3)){
    IM=((CNEM+CNEMAP)/NEM);
    RE=((CDMIEQ-IM)*NEG);
} else if ((IONS==0) && (animal->SEX == 3)){
    IM=((CNEM+CNEMAP)/(NEM*1.12));
} else if ((IONS==0) && (animal->SEX > 3)){
    IM=((CNEM+CNEMAP)/(NEM*1.12));
} else if ((IONS==1) && (animal->SEX < 3)){
    IM=((CNEM+CNEMAP)/(NEM*1.12));
} else if ((IONS==1) && (animal->SEX == 3)){
    IM=((CNEM+CNEMAP)/(NEM*1.12));
} else if ((IONS==1) && (animal->SEX > 3)){
    IM=((CNEM+CNEMAP)/(NEM*1.12));
} if (animal->SEX < 3){
    RE=((CDMIEQ-IM)*NEG);
} else if (animal->SEX ==3){
    RE=((CPDVI-IM)*NEG);
} else if (animal->SEX > 3){
    RE=((SPDVI-IM)*NEG);
}
if ((animal->AGE <= 30) && (animal->SEX<3)){
    KM=NEM/ME;
    InternInsul=2.5;
    ExternInsul= ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
    Insulate= InternInsul + ExternInsul;
    SurArea=(.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
    MEIntake=PDMI * ME;
    HeatProd= (MEIntake - RE)/SurArea;
    LowCritTemp= 39 - (Insulate * ((HeatProd/SurArea) * .85));
    CNEMCS= (KM* ((SurArea *(LowCritTemp - Tc))/Insulate));
}
else if ((animal->AGE > 30) && (animal->AGE <= 365) &&
         (animal->SEX<3)){
    KM=NEM/ME;
    InternInsul=6.5;
    ExternInsul= ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
    Insulate= InternInsul + ExternInsul;
    SurArea=(.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
    MEIntake=PDMI * ME;
    HeatProd= (MEIntake - RE)/SurArea;
    LowCritTemp= 39 - (Insulate * ((HeatProd/SurArea) * .85));
    CNEMCS= (KM* ((SurArea *(LowCritTemp - Tc))/Insulate));
}
else if ((animal->AGE > 365) && (animal->AGE <= 730) &&
         (animal->SEX<3)){
    KM=NEM/ME;
    InternInsul=5.1875 + (.3125 * animal->BCS);
    ExternInsul= ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
    Insulate= InternInsul + ExternInsul;
    SurArea=(.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
    MEIntake=PDMI * ME;
    HeatProd= (MEIntake - RE)/SurArea;
    LowCritTemp= 39 - (Insulate * ((HeatProd/SurArea) * .85));
    CNEMCS= (KM* ((SurArea *(LowCritTemp - Tc))/Insulate));
}
else if ((animal->AGE > 730) && (animal->SEX<3)){
    KM=NEM/ME;
    InternInsul=5.25 + (.75 * animal->BCS);
    ExternInsul= ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
    Insulate= InternInsul + ExternInsul;
    SurArea=(.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
    MEIntake=PDMI * ME;
    HeatProd= (MEIntake - RE)/SurArea;
    LowCritTemp= 39 - (Insulate * ((HeatProd/SurArea) * .85));
    CNEMCS= (KM* ((SurArea *(LowCritTemp - Tc))/Insulate));
}
else if ((animal->AGE > 365) && (animal->AGE <= 730) &&
         (animal->SEX==3)){
    KM=NEM/ME;
    InternInsul=5.1875 + (.3125 * animal->BCS);
    ExternInsul= ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
Insulate = InternInsul + ExternInsul;
SurArea = (.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
MEIntake = CPDMI * ME;
HeatProd = (MEIntake - RE)/SurArea;
LowCritTemp = 39 - (Insulate * ((HeatProd/SurArea) * .85));
CNEMCS = (KM* ((SurArea *(LowCritTemp - Tc))/Insulate));
}
else if ((animal->AGE > 730) && (animal->SEX==3)){
  KM = NEM/ME;
  InternInsul = 5.25 + (.75 * animal->BCS);
  ExternInsul = ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
  Insulate = InternInsul + ExternInsul;
  SurArea = (.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
  MEIntake = CPDMI * ME;
  HeatProd = (MEIntake - RE)/SurArea;
  LowCritTemp = 39 - (Insulate * ((HeatProd/SurArea) * .85));
  CNEMCS = (KM* ((SurArea *(LowCritTemp - Tc))/Insulate));
}
else if ((animal->AGE > 30) && (animal->AGE <= 365) &&
  (animal->SEX>3)){
  KM = NEM/ME;
  InternInsul = 6.5;
  ExternInsul = ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
  Insulate = InternInsul + ExternInsul;
  SurArea = (.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
  MEIntake = SPDMI * ME;
  HeatProd = (MEIntake - RE)/SurArea;
  LowCritTemp = 39 - (Insulate * ((HeatProd/SurArea) * .85));
  CNEMCS = (KM* ((SurArea *(LowCritTemp - Tc))/Insulate));
}
else if ((animal->AGE > 365) && (animal->AGE <= 730) &&
  (animal->SEX>3)){
  KM = NEM/ME;
  InternInsul = 5.1875 + (.3125 * animal->BCS);
  ExternInsul = ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
  Insulate = InternInsul + ExternInsul;
  SurArea = (.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
  MEIntake = SPDMI * ME;
  HeatProd = (MEIntake - RE)/SurArea;
  LowCritTemp = 39 - (Insulate * ((HeatProd/SurArea) * .85));
  CNEMCS = (KM* ((SurArea *(LowCritTemp - Tc))/Insulate));
}
else if ((animal->AGE > 730) && (animal->SEX>3)){
  KM = NEM/ME;
  InternInsul = 5.25 + (.75 * animal->BCS);
  ExternInsul = ((7.36 - (.296 * WIND) + (2.55* HAIR)) * HIDE);
  Insulate = InternInsul + ExternInsul;
  SurArea = (.09 * (pow((animal->SBWTK - animal->GUWTK), .67)));
  MEIntake = SPDMI * ME;
  HeatProd = (MEIntake - RE)/SurArea;
  LowCritTemp = 39 - (Insulate * ((HeatProd/SurArea) * .85));
}
CNEMCS = (KM * ((SurArea * (LowCritTemp - Tc))/Insulate));
}
if (Tc < 25) {
  if (((IONO==0) & (animal->SEX < 3))){
    IM = ((CNEM + CNEMAP + CNEMCS) / NEM);
  }
  else if (((animal->SEX==3) & (animal->BCS>5) & (IONO==0))){
    IM = ((CNEM + CNEMAP + CNEMCS + (animal->YEN) + CNEMP) / NEM);
  }
  else if (((IONO==0) & (animal->SEX == 3) & (animal->BCS<=5))){
    IM = ((CNEM + CNEMAP + (.525 * CNEMCS)) + (animal->YEN) + (CNEMP)) / NEM;
  }
  else if (((IONO==0) & (animal->SEX > 3))){
    IM = ((CNEM + CNEMAP + CNEMCS) / NEM);
  }
  else if (((animal->SEX==3) & (animal->BCS>5) & (IONO==1))){
    IM = ((CNEM + CNEMAP + CNEMCS + (animal->YEN) + CNEMP) / (NEM*1.12));
  }
  else if (((IONO==1) & (animal->SEX < 3))){
    IM = ((CNEM + CNEMAP + CNEMCS) / (NEM*1.12));
  }
  else if (((IONO==1) & (animal->SEX == 3) & (animal->BCS<=5))){
    IM = ((CNEM + CNEMAP + (.479 * CNEMCS)) + (animal->YEN) + (CNEMP)) / (NEM*1.12));
  }
  else if (((IONO==1) & (animal->SEX > 3))){
    IM = ((CNEM + CNEMAP + CNEMCS) / (NEM*1.12));
  }
}
else if ((Tc >= 25) & (Tc < 30)){
  IM = IM * 1.07;
}
else if (Tc>=30){
  IM = IM * 1.18;
}
if (animal->SEX < 3){
  RE = ((CDMIEQ - IM) * NEG);
}
else if (animal->SEX ==3){
  RE = ((CPDMI - IM) * NEG);
}
else if (animal->SEX > 3){
  RE = ((SPDMI - IM) * NEG);
}
if ((RE >= 0) & (animal->SEX == 0)){
  SBWTG = (13.91* pow(RE, .9116) * (pow(tempEQSBW, -.6837))) * 0.95215;
  if (SBWTG > (((double) animal->PADG) * .453599 * .96 * 0.95215))

}
} else if (SBWTG<=(((double)animal->PADG) * .453599 * 0.95215))
) {animal->SBWTK+=SBWTG;
}
else if ((RE < 0) && (animal->SEX == 0)) {
SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEQSBW, -.6837))) * 0.95215 *(-1);
if (SBWTG>(((double) animal->PADG) * .453599 *.96 * 0.95215))
) {
( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96 *
0.95215) );
}
else if (SBWTG<=(((double)animal->PADG) * .453599 * 0.95215))
) {animal->SBWTK+=SBWTG;
}
else if ((RE >= 0) && (animal->SEX == 3) && (animal->PREG == 0)) {
SBWTG=(13.91*pow(RE,.9116)*(pow(tempEQSBW, -.6837))) * 0.95215;
if (SBWTG>(((double) animal->PADG) * .453599 *.96 * 0.95215))
) {
( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96 *
0.95215) );
}
else if (SBWTG<=(((double)animal->PADG) * .453599 * 0.95215))
) {animal->SBWTK+=SBWTG;
}
else if ((RE < 0) && ((animal->SEX == 3) && (animal->PREG == 0))) {
SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEQSBW, -.6837))) * 0.95215 *(-1);
if (SBWTG>(((double) animal->PADG) * .453599 *.96 * 0.95215))
) {
( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96 *
0.95215) );
}
else if (SBWTG<=(((double)animal->PADG) * .453599 * 0.95215))
) {animal->SBWTK+=SBWTG;
}
else if ((RE >= 0) && ((animal->SEX == 3) && (animal->PREG == 0)))
) {animal->SBWTK+=SBWTG;
}
}
if (calcuADG == 1) && (animal->SEX > 3) )
if ((RE >= 0) && (animal->SEX == 4))
SBWTG=(13.91*pow(RE,.9116)*(pow(tempEQSBW, -.6837))) * 
0.95215;
if (SBWTG>(((double) animal->PADG) * .453599 *.96 * 0.95215) 
)
( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96 * 
0.95215) + GUWTKD) );
else if (SBWTG<=(((double)animal->PADG) * .453599 *.96 * 
0.95215))
animal->SBWTK+=(SBWTG + GUWTKD);
}
else if ((RE < 0) && (animal->SEX==3) && (animal->PREG == 
1)))
SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEQSBW, -.6837))) * 
0.95215 *(-1); if (SBWTG>((double) animal->PADG) * .453599 *.96 * 0.95215) 
)
( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96 * 
0.95215) + GUWTKD) );
else if (SBWTG<=(((double)animal->PADG) * .453599 *.96 * 
0.95215))
animal->SBWTK+=(SBWTG + GUWTKD);
}
else if ((RE > 0) && (animal->SEX==2) )
SBWTG=(13.91*pow(RE,.9116)*(pow(tempEQSBW, -.6837))); if (SBWTG>((double) animal->PADG) * .453599 *.96) )
( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96) );
else if (SBWTG<=(((double)animal->PADG) * .453599 *.96) )
animal->SBWTK+=SBWTG;
}
else if ((RE < 0) && (animal->SEX==2) )
SBWTG=(13.91*pow((-RE),.9116)*pow(tempEQSBW, -.6837))*(-
1)*1.23;
if (SBWTG>((double) animal->PADG) * .453599 *.96) )
( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96) );
else if (SBWTG<=(((double)animal->PADG) * .453599 *.96) )
animal->SBWTK+=SBWTG;
}
( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96 * 0.95215) );
}
else if (SBWTG<=(((double)animal->PADG) * .453599 * 0.95215)) {
    animal->SBWTK+=SBWTG;
}
else if ((RE < 0) && (animal->SEX == 4)) {
    SBWTG=(13.91*pow((-RE), .9116)*(pow(tempEQSBW, -.6837))) * 0.95215 *(-1);
    if (SBWTG>(((double) animal->PADG) * .453599 *.96 * 0.95215)) {
        ( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96 * 0.95215) );
    }
    else if (SBWTG<=(((double)animal->PADG) * .453599 * 0.95215)) {
        animal->SBWTK+=SBWTG;
    }

else if ((RE > 0) && (animal->SEX==5)) {
    SBWTG=(13.91*pow(RE, .9116)*(pow(tempEQSBW, -.6837)));  
    if (SBWTG>(((double) animal->PADG) * .453599 *.96)) {
        ( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96) );
    }
    else if (SBWTG<=(((double)animal->PADG) * .453599 * .96)) {
        animal->SBWTK+=SBWTG;
    }
}
else if ((RE < 0) && (animal->SEX==5)) {
    SBWTG=(13.91*pow((-RE), .9116)*(pow(tempEQSBW, -.6837)))*(-1)*1.23;
    if (SBWTG>(((double) animal->PADG) * .453599 *.96)) {
        ( animal->SBWTK+=(((double)animal->PADG) * .453599 *.96) );
    }
    else if (SBWTG<=(((double)animal->PADG) * .453599 * .96)) {
        animal->SBWTK+=SBWTG;
    }
}
if ((calcuADG == 0) && (animal->SEX > 3) && (calcusea- 
sADG==0)) {
    if ((RE >= 0) && (animal->SEX == 4)) {
        SBWTG=(13.91*pow(RE, .9116)*(pow(tempEQSBW, -.6837)))* 0.95215;
        if (SBWTG>(((double) animal->UADG) * .453599 *.96 * 0.95215)) {
            ( animal->SBWTK+=(((double)animal->UADG) * .453599 *.96 * 0.95215) );
        }
else if (SBWTG <= ((double)animal->UADG) * 0.453599 * 0.95215)
    {animal->SBWTK += SBWTG;}
}
else if ((RE < 0) && (animal->SEX == 4))
    {SBWTG = (13.91 * pow(-RE, 0.9116) * pow(tempEQSBW, -0.6837)) * 0.95215 * (-1);
     if (SBWTG > ((double) animal->UADG) * 0.453599 * 0.96 * 0.95215)
        {animal->SBWTK += ((double) animal->UADG) * 0.453599 * 0.96 * 0.95215;}
    }
else if (SBWTG <= ((double)animal->UADG) * 0.453599 * 0.96)
    {animal->SBWTK += SBWTG;}
}
else if ((RE > 0) && (animal->SEX == 5))
    {SBWTG = (13.91 * pow(RE, 0.9116) * pow(tempEQSBW, -0.6837));
     if (SBWTG > ((double) animal->UADG) * 0.453599 * 0.96)
        {animal->SBWTK += ((double) animal->UADG) * 0.453599 * 0.96;}
    }
else if (SBWTG <= ((double)animal->UADG) * 0.453599 * 0.96)
    {animal->SBWTK += SBWTG;}
}
else if ((RE < 0) && (animal->SEX == 5))
    {SBWTG = (13.91 * pow(-RE, 0.9116) * pow(tempEQSBW, -0.6837)) * (-1) * 1.23;
     if (SBWTG > ((double) animal->UADG) * 0.453599 * 0.96)
        {animal->SBWTK += ((double) animal->UADG) * 0.453599 * 0.96;}
    }
else if (SBWTG <= ((double)animal->UADG) * 0.453599 * 0.96)
    {animal->SBWTK += SBWTG;}
}
if ((SBWTG > animal->UADG))
    {excessSBWTG = SBWTG - (animal->UADG);
     if ((excessSBWTG > 0) && (animal->SEX == 4))
        {extraRE = pow(((excessSBWTG/(pow(tempEQSBW, -0.6837)))/0.95215)/13.91, (1/0.9116));
         excessSPDMI = extraRE/NEG;
         if (excessSPDMI > 0)
             {SPDMIS -= excessSPDMI;}}
     else if ((excessSBWTG > 0) && (animal->SEX == 5))
        {extraRE = pow(((excessSBWTG/(pow(tempEQSBW, -0.6837)))/13.91, (1/0.9116));
excessSPDMI= extraRE/NEG;
if ((excessSPDMI>0) & (SPDMIS>excessSPDMI)){
    TSPDMIS=excessSPDMI;
} else if ((excessSPDMI > 0) & (SPDMIS<excessSPDMI)){
    TSPDMIS=SPDMIS;
}
}
}
}
}
else if ((calcuADG == 0) & (animal->SEX > 3) & (calcusea-
sADG==1)){
    if ((currentDATE >= 1) & (currentDATE <91) ){
        if ((RE >= 0) & (animal->SEX == 4)){
            SBWTG=(13.91*pow(RE,.9116)*(pow(tempEQSBW, -.6837))) * 0.95215;
            if (SBWTG>((double) animal->UADG1) * .453599 *.96 * 0.95215) ){
                animal->SBWTK+=(double)animal->UADG1) * .453599 *.96 * 0.95215);
            }
        } else if (SBWTG<=((double)animal->UADG1) * .453599 *
0.95215) ){
            animal->SBWTK+=SBWTG;
        }
    }
    else if ((RE < 0) & (animal->SEX == 4)){
        SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEQSBW, -.6837))) * 0.95215 *(-1);
        if (SBWTG>((double) animal->UADG1) * .453599 *.96 * 0.95215) ){
            animal->SBWTK+=(double)animal->UADG1) * .453599 *.96 * 0.95215);
        }
    } else if (SBWTG<=((double)animal->UADG1) * .453599 *
0.95215) ){
        animal->SBWTK+=SBWTG;
    }
    else if ((RE > 0) & (animal->SEX==5)){
        SBWTG=(13.91*pow(RE,.9116)*(pow(tempEQSBW, -.6837)));
        if (SBWTG>((double) animal->UADG1) * .453599 *.96 ){
            animal->SBWTK+=(double)animal->UADG1) * .453599 *.96 );
        }
    } else if (SBWTG<=((double)animal->UADG1) * .453599 *.96 ){
        animal->SBWTK+=SBWTG;
    }
    else if ((RE < 0) & (animal->SEX==5)){
        SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEQSBW, -.6837)))*(-1)*1.23;
if (SBWTG>((double) animal->UADG1) * .453599 *.96) {
    (animal->SBWTK+=((double)animal->UADG1) * .453599 *.96) ;
} else if (SBWTG<=((double)animal->UADG1) * .453599 *.96) {
    animal->SBWTK+=SBWTG;
}

if ((SBWTG> animal->UADG)){
    excessSBWTG=SBWTG - (animal->UADG);
    if ((excessSBWTG > 0) && (animal->SEX==4)){
        extraRE= pow((((excessSBWTG/(pow(tempEQSBW, -.6837)))/0.95215)/13.91), (1/.9116));
        excessSPDMI= extraRE/NEG;
        if (excessSPDMI>0){
            SPDMIS-=excessSPDMI;
        }
    } else if ((excessSBWTG > 0) && (animal->SEX==5)){
        extraRE= pow(((excessSBWTG/(pow(tempEQSBW, -.6837)))/13.91), (1/.9116));
        excessSPDMI= extraRE/NEG;
        if ((excessSPDMI>0) && (SPDMIS>excessSPDMI)){
            TSPDMIS-=excessSPDMI;
        } else if ((excessSPDMI > 0) && (SPDMIS<excessSPDMI)){
            TSPDMIS-=SPDMIS;
        }
    }
} else if ((currentDATE >= 91) && (currentDATE <182) ){
    if ((RE >= 0) && (animal->SEX == 4)){
        SBWTG=(13.91*pow(RE,.9116)*(pow(tempEQSBW, -.6837))) * 0.95215;
        if (SBWTG>(((double) animal->UADG2) * .453599 *.96 *
            0.95215) ){
            (animal->SBWTK+=(((double)animal->UADG2) * .453599 *.96 *
                0.95215) );
        } else if (SBWTG<=(((double)animal->UADG2) * .453599 *
            0.95215) ){
            animal->SBWTK+=SBWTG;
        }
    } else if ((RE < 0) && (animal->SEX == 4)){
        SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEQSBW, -.6837))) * 0.95215 *(-1);
        if (SBWTG>(((double) animal->UADG2) * .453599 *.96 *
            0.95215) ){
            (animal->SBWTK+=(((double)animal->UADG2) * .453599 *.96 *
                0.95215) );
        } else if (SBWTG<=(((double)animal->UADG2) * .453599 *
            0.95215) ){
            animal->SBWTK+=SBWTG;
        }
    }
}
else if (SBWTG <= (((double) animal->UADG2) * 0.453599 * 0.95215)) {
    animal->SBWTK += SBWTG;
}
else if ((RE > 0) && (animal->SEX == 5)) {
    SBWTG = (13.91 * pow(RE, .9116) * (pow(tempEQSBW, -.6837))); 
    if (SBWTG > (((double) animal->UADG2) * .453599 * .96)) {
        (animal->SBWTK += (((double) animal->UADG2) * .453599 * .96));
    }
    else if (SBWTG <= (((double) animal->UADG2) * .453599 * .96)) {
        animal->SBWTK += SBWTG;
    }
}
else if ((RE < 0) && (animal->SEX == 5)) {
    SBWTG = (13.91 * pow((-RE), .9116) * (pow(tempEQSBW, -.6837))) * (-1) * 1.23;
    if (SBWTG > (((double) animal->UADG2) * .453599 * .96)) {
        (animal->SBWTK += (((double) animal->UADG2) * .453599 * .96));
    }
    else if (SBWTG <= (((double) animal->UADG2) * .453599 * .96)) {
        animal->SBWTK += SBWTG;
    }
}
if ((SBWTG > animal->UADG2)) {
    excessSBWTG = SBWTG - (animal->UADG);
    if ((excessSBWTG > 0) && (animal->SEX == 4)) {
        extraRE = pow(((excessSBWTG/(pow(tempEQSBW, -.6837)))/0.95215)/13.91, (1/.9116));
        excessSPDMI = extraRE/NEG;
        if (excessSPDMI > 0) {
            SPDMIS -= excessSPDMI;
        }
    } else if ((excessSBWTG > 0) && (animal->SEX == 5)) {
        extraRE = pow(((excessSBWTG/(pow(tempEQSBW, -.6837)))/0.95215)/13.91, (1/.9116));
        excessSPDMI = extraRE/NEG;
        if ((excessSPDMI > 0) && (SPDMIS > excessSPDMI)) {
            TSPDMIS -= excessSPDMI;
        } else if (excessSPDMI > 0) && (SPDMIS < excessSPDMI) {
            TSPDMIS -= SPDMIS;
        }
    }
}
else if ((currentDATE >= 182) && (currentDATE < 274)) {
    if ((RE >= 0) && (animal->SEX == 4)) {
        SBWTG = (13.91 * pow(RE, .9116) * (pow(tempEQSBW, -.6837)) *
0.95215;
if (SBWTG>(((double) animal->UADG3) *.453599 *.96 * 0.95215) ){  
  ( animal->SBWTK+=(((double)animal->UADG3) *.453599 *.96 * 0.95215) );
} else if (SBWTG<=(((double)animal->UADG3) *.453599 * 0.95215) ){
  animal->SBWTK+=SBWTG;
}
else if ( (RE < 0) && (animal->SEX == 4)){
  SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEQSBW, -.6837))) * 0.95215 *(-1);
  if (SBWTG>(((double) animal->UADG3) *.453599 *.96 * 0.95215) ){  
    ( animal->SBWTK+=(((double)animal->UADG3) *.453599 *.96 * 0.95215) );
  }
else if (SBWTG<=(((double)animal->UADG3) *.453599 *.96 )){
  animal->SBWTK+=SBWTG;
}
else if ( (RE > 0) && (animal->SEX==5)){
  SBWTG=(13.91*pow(RE,.9116)*(pow(tempEQSBW, -.6837)));
  if (SBWTG>(((double) animal->UADG3) *.453599 *.96 )}{  
    ( animal->SBWTK+=(((double)animal->UADG3) *.453599 *.96 ) );
  }
else if (SBWTG<=(((double)animal->UADG3) *.453599 *.96 )){
  animal->SBWTK+=SBWTG;
}
else if ( (RE < 0) && (animal->SEX==5)){
  SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEQSBW, -.6837)))*(-1)*1.23;
  if (SBWTG>(((double) animal->UADG3) *.453599 *.96 ) ){  
    ( animal->SBWTK+=(((double)animal->UADG3) *.453599 *.96 ) );
  }
else if (SBWTG<=(((double)animal->UADG3) *.453599 *.96 )){
  animal->SBWTK+=SBWTG;
}
}
if((SBWTG> animal->UADG3)){
excessSBWTG=SBWTG - (animal->UADG);  
if((excessSBWTG > 0) && (animal->SEX==4)){
  extraRE= pow(((excessSBWTG/(pow(tempEQSBW, -.6837)))/0.95215)/13.91),1/.9116));
excessSPDMI= extraRE/NEG;  
if(excessSPDMI>0){
  SPDMIS=excessSPDMI;
else if ((excessSBWTG > 0) && (animal->SEX==5)){
    extraRE= pow(((excessSBWTG/(pow(tempEFSBW, -.6837))))/13.91), (1/1.9116));
    excessSPDMI= extraRE/NEG;
    if ((excessSPDMI>0) && (SPDMIS>excessSPDMI)){
        TSPDMIS-=excessSPDMI;
    } else if ((excessSPDMI > 0) && (SPDMIS<excessSPDMI)){
        TSPDMIS-=SPDMIS;
    }
}

else if ((currentDATE >= 274) && (currentDATE <=365) ){
    if ((RE >= 0) && (animal->SEX == 4)){
        SBWTG=(13.91*pow(RE,.9116)*(pow(tempEFSBW, -.6837))) * 0.95215;
        if (SBWTG>(((double) animal->UADG4) * .453599 *.96 * 0.95215) ){
            animal->SBWTK+=(((double)animal->UADG4) * .453599 *.96 * 0.95215) ;
        } else if (SBWTG<=(((double)animal->UADG4) * .453599 * .96 ) {
            animal->SBWTK+=SBWTG;
        }
    } else if ((RE < 0) && (animal->SEX == 4)){
        SBWTG=(13.91*pow((-RE),.9116)*(pow(tempEFSBW, -.6837))) * 0.95215 *(-1);
        if (SBWTG>(((double) animal->UADG4) * .453599 *.96 * 0.95215) ){
            animal->SBWTK+=(((double)animal->UADG4) * .453599 *.96 * 0.95215) ;
        } else if (SBWTG<=(((double)animal->UADG4) * .453599 * .96 ) {
            animal->SBWTK+=SBWTG;
        }
    } else if ((RE > 0) && (animal->SEX==5)){
        SBWTG=(13.91*pow(RE,.9116)*(pow(tempEFSBW, -.6837)));
        if (SBWTG>(((double) animal->UADG4) * .453599 *.96) ){
            animal->SBWTK+=(((double)animal->UADG4) * .453599 *.96) ;
        } else if (SBWTG<=(((double)animal->UADG4) * .453599 * .96 ) {
            animal->SBWTK+=SBWTG;
        }
    }
else if ((RE < 0) && (animal->SEX==5)){
    SBWTG=(13.91*pow((-RE), .9116)*(pow(tempEQSBW, -.6837)))*(-1)*1.23;
    if (SBWTG>(((double) animal->UADG4) * .453599 *.96) ){
        ( animal->SBWTK+=(((double)animal->UADG4) * .453599 *.96) );
    } else if (SBWTG<=(((double)animal->UADG4) * .453599 * .96) ){animal->SBWTK+=SBWTG;
    }
}
if ((SBWTG> animal->UADG2)){
    excessSBWTG=SBWTG - (animal->UADG);
    if ((excessSBWTG > 0) && (animal->SEX==4)){
        extraRE= pow((((excessSBWTG/(pow(tempEQSBW, -.6837)))/0.95215)/13.91), (1/.9116));
        excessSPDMI= extraRE/NEG;
        if (excessSPDMI>0){
            SPDMIS-=excessSPDMI;
        }
    } else if ((excessSBWTG > 0) && (animal->SEX==5)){
        extraRE= pow(((excessSBWTG/(pow(tempEQSBW, -.6837)))/13.91), (1/.9116));
        excessSPDMI= extraRE/NEG;
        if ((excessSPDMI>0) && (SPDMIS>excessSPDMI)){
            TSPDMIS-=excessSPDMI;
        } else if ((excessSPDMI > 0) && (SPDMIS<excessSPDMI)){
            TSPDMIS-=SPDMIS;
        }
    }
}
animal->CDWTK=(animal->SBWTK/.96);
animal->CDWT=(animal->CDWTK * 2.20459);
animal->EBWT=(animal->SBWTK *.891);
animal->EQSBW=(animal->SBWTK * (478/((animal->PMWT*.453599) *.96)));
animal->EQEBW=(animal->EQSBW *.891);
return;
}
void Nutrition::ResetNutritionVariables(){
    APDMIF=0;
    ASPDMIF=0;
    TPDMIF=0;
    TSPDMIF=0;
TPDMIFA=0;
TSPDMIFA=0;
TPDMIS=0;
TCPDMIS=0;
TSPDMIS=0;
hayAVAIL=0;
hayConsump=0;
Concentrate=0;
hayConsumpS=0;
ConcentrateS=0;
}
AppWizard has created this HerdSim application for you. This application not only demonstrates the basics of using the Microsoft Foundation classes but is also a starting point for writing your application. This file contains a summary of what you will find in each of the files that make up your HerdSim application.

HerdSim.h
This is the main header file for the application. It includes other project specific headers (including Resource.h) and declares the CHerdSimApp application class.

HerdSim.cpp
This is the main application source file that contains the application class CHerdSimApp.

HerdSim.rc
This is a listing of all of the Microsoft Windows resources that the program uses. It includes the icons, bitmaps, and cursors that are stored in the RES subdirectory. This file can be directly edited in Microsoft Developer Studio.

res\HerdSim.ico
This is an icon file, which is used as the application's icon. This icon is included by the main resource file HerdSim.rc.

res\HerdSim.rc2
This file contains resources that are not edited by Microsoft Developer Studio. You should place all resources not editable by the resource editor in this file.

HerdSim.clw
This file contains information used by ClassWizard to edit existing classes or add new classes. ClassWizard also uses this file to store information needed to create and edit message maps and dialog data.
maps and to create prototype member functions.

AppWizard creates one dialog class:

HerdSimDlg.h, HerdSimDlg.cpp - the dialog
These files contain your CHerdSimDlg class. This class defines the behavior of your application's main dialog. The dialog's template is in HerdSim.rc, which can be edited in Microsoft Developer Studio.

StdAfx.h, StdAfx.cpp
These files are used to build a precompiled header (PCH) file named HerdSim.pch and a precompiled types file named StdAfx.obj.

Resource.h
This is the standard header file, which defines new resource IDs. Microsoft Developer Studio reads and updates this file.

AppWizard uses "TODO:" to indicate parts of the source code you should add to or customize.

If your application uses MFC in a shared DLL, and your application is in a language other than the operating system's current language, you will need to copy the corresponding localized resources MFC40XXX.DLL from the Microsoft Visual C++ CD-ROM onto the system or system32 directory, and rename it to be MFCLOC.DLL. ("XXX" stands for the language abbreviation. For example, MFC40DEU.DLL contains resources translated to German.) If you don't do this, some of the UI elements of your application will remain in the language of the operating system.
// check if memory leaks
#ifdef _DEBUG
CMemoryState oldMemState, newMemState, diffMemState;
oldMemState.Checkpoint();
#endif

// do your memory allocations and deallocations...
CString s = "This is a frame variable";
// the next object is a heap object
CPerson* p = new CPerson( "Smith", "Alan", "581-0215" );

#ifdef _DEBUG
newMemState.Checkpoint();
if( diffMemState.Difference( oldMemState, newMemState ) )
{
    TRACE( "Memory leaked!\n" );
    diffMemState.DumpStatistics(); // call DumpStatistics to get
    information about the objects that have not been deallocated:
}
#endif
StdAfx.cpp
//stdafx.cpp : source file that includes just the
// standard includes
//HerdSim.pch will be the pre-compiled header
//stdafx.obj will contain the pre-compiled type
//information

#include "stdafx.h"
#include <afx.h>
Tools.cpp
#include "stdafx.h"
#include "HerdSim.h"
#include "HerdSimDlg.h"
#include <iostream.h>
#include <math.h>
#include <stdlib.h>
#include <afx.h>
#include "tools.h"

int Julian(int Month, int Day)
{
    int Correction=0;
    int JulianValue;
    if ( (Month < 1) || (Month > 12) )
    {
        cout << "There is a problem with the Month value." << endl;
    }
    switch (Month){
    case (2):
    case (6):
    case (7): Correction= 1; break;
    case (3): Correction=-1; break;
    case (8): Correction= 2; break;
    case (9):
    case (10):Correction=3; break;
    case (11):
    case (12):Correction=4; break;
    }
    JulianValue = ( ( (Month-1) * 30) + Correction + Day);
    return ( JulianValue );
}

double GetStochasticNumber(double Deviation, double Average) /*Returns a value from a
normal distribution.*/
{
    double z,z1,z2;
    double mcx;
    z1=( rand() / (double)RAND_MAX); //Initial Randomization
    z2=( rand() / (double)RAND_MAX); //Initial Randomization
    z=cos(6.2831853*z2)*sqrt(-2.0*log(z1)); //Calculated Z value
    z=z * Deviation; //Turned into Standard Deviations
    mcx = Average + z; //Added to the Average
    return mcx;
}

int GetRandom(){
    int i;
    for (i=0; i < 10000; i++){
        rand();
    }
}
return rand();
}

void cleardevice()
{
    cout << endl << endl << endl << endl << endl;
    cout << endl << endl << endl << endl << endl;
    cout << endl << endl << endl << endl << endl;
    cout << endl << endl << endl << endl << endl;
    cout << endl << endl << endl << endl << endl;
    cout << endl << endl << endl << endl << endl;
}
Header Files
#ifndef COW_H
#define COW_H
#include <iostream.h>
#include <fstream.h>

typedef struct AnimalRecord {
    int ID;
    char SEX;
    int AGE;
    int BIRTHDATE;
    int WEANDATE;
    int WEAN;
    int CALFSEAS;
    char BREED;
    char COLR;
    char PURE;
    char REPLA;
    char PREG;
    char SERVI;
    char PARIE;
    char UNPREG;
    char BCS;
    double CDWT;
    double PMWT;
    double PMSWT;
    double PADG;
    double UADG;
    double UADG1;
    double UADG2;
    double UADG3;
    double UADG4;
    int PUBER;
    int INELIG;
    double CUCYC;
    double CNDAT;
    int DPREG;
    double PGEST;
    int FIDN;
    char CSEX;
    int CBIRTHDATE;
    double GUWTK;
    double CBWT;
    int CIDN;
    int MIDN;
    double CADAT;
    double PPI;
    char LACT;
    double LDAY;
    double YN;
    double YEN;
    unsigned int AGED;
};
char DIE;
char CULL;
char DETEC;
char DYSTO;
char Fgeneration
char SIREBREED;
char DAMBREED;
char GSIREBREED;
char GDAMBREED;
char GGSIREBREED;
char GGDAMBREED;
char GGGSIREBREED;
char GGGDAMBREED;
double BWT;
double CDWTK;
double EBWTK;
double EQSBW;
double EQEBW;
double SBWTK;
double WEANWT;
char IMPLANT;
char PAST;
char SOUR;
char Puberty;
char SEAOP;
char SEAOP1;
char SEAOP2;
char FAIL;
char FWEAN;
char SellCalf;
double MCMFC;
double MCSNF;
double PGAVG;
double PGSD;
double DHETADG;
double DHETBWT;
double DHETMWT;
double MHETADG;
double MHETBWT;
double ACBWT;
AnimalRecord *prev;
AnimalRecord *next;
};

class Cow{
private:
char SEASON;
char PURE;
char COLR;
char CALFEASE;
char EZCALF;
char EZBREED;
char HBREED;
char HBBREED;
char BuyPure;
char BuyBreed;
int BULBREED[13];
double TCDAT;
double NCTB;
char TEST;
char BUYP;
char BUYRC;
char SelectWIMer;
int CLNUM;
int NHFC;
int NSTC;
int NSHFC;
int NSSTC;
int REPLF;
int TNCW;
double TWHS;
double TWSS;
double WNWT;
int REPLR;
int TSTOP;
double AWNWT;
double BSAL;
int earlyday;
int earlymonth;
int fdbmsday;
int fdbmsmonth;
int earlyday2;
int earlymonth2;
int fdbmsday2;
int fdbmsmonth2;
int weansmonth;
int weansday;
int weanfmonth;
int weanfday;
int Servi;
double PER[13];

AnimalRecord *herdhead;
private:
void UseInput1();
void UseInput1A();
void UseInput1B();
public:
void UseInput11();
void ResetCowConCntr();
int GetOriginalNumOfCows();
int UseYearOption(int);
void Init();
Cow();
void DumpList(ofstream&);
void ShrinkAnimal(AnimalRecord *);
void Lactation(AnimalRecord *);
void ChangeSex(AnimalRecord *);
void PregnancyTest(AnimalRecord *);
void Puberty(AnimalRecord *);
void Sterility(AnimalRecord *);
void Dystocia(AnimalRecord *);
void BreedFemales(AnimalRecord *);
void Parturition(AnimalRecord *);
void Mortality(AnimalRecord *);
void SBBreeders(AnimalRecord *);
void Wean(AnimalRecord *);
void ReproStats();
void CullCows(AnimalRecord *);
void BirthAnimal (AnimalRecord *, AnimalRecord *);
void DeleteAnimal (int);
int DeadCattle(AnimalRecord *);
void DeleteHerd();
void CreateHerd();
void Abort(AnimalRecord *);
void SortList(int);
void ResetCowVariables();
AnimalRecord* GetAnimal (int ID);
AnimalRecord* GetMother (int ID);
AnimalRecord* GetHeadAnimal ();
int LastID();
void AddAnimal (AnimalRecord*);
void CalfMilk (AnimalRecord*);
AnimalRecord* GetSex(int, int);
};
#endif
Forage.h
#ifndef FORAGE_H
#define FORAGE_H

class Forage{
    double SLOE;
    int SREG;
    double FR[22];
    int dateInMonth[7];
    int dateInDay[7];
    int dateOutMonth[7];
    int dateOutDay[7];
    int dateRInMonth[7];
    int dateRInDay[7];
    int dateROutMonth[7];
    int dateROutDay[7];
    int TPAS[7];
    int HLAND[7];
    int WLAND[7];
    int TRPAS[7];
    int HRLAND[7];
    int WRLAND[7];
    int pasture;
    int rpasture;
    double TAPG;
    double APRILGROWTH;
    double AUGUSTGROWTH;
    double DECEMBERGROWTH;
    double FEBRUARYGROWTH;
    double JANUARYGROWTH;
    double JULYGROWTH;
    double JUNEGROWTH;
    double MARCHGROWTH;
    double MAYGROWTH;
    double NOVEMBERGROWTH;
    double OCTOBERGROWTH;
    double SEPTEMBERGROWTH;
    double RAINAPRIL;
    double RAINAUGUST;
    double RAINDECEMBER;
    double RAINFEBRUARY;
    double RAINJANUARY;
    double RAINJULY;
    double RAINJUNE;
    double RAINMARCH;
    double RAINMAY;
    double RAINNOVEMBER;
    double RAINOCTOBER;
    double RAINSEPTEMBER;
    int EROS;
    int FERT;
    int GRAZ;

#endif
int WEDS;
double PREP;

private:
void UseInput5();
void UseInput6();
public:
void ResetForageConCntr();
void UseInput10();
void UseInput9();
Forage();
void Init();
int Temperature();
void MainForage();
void ResetForageVariables();

};
#endif
HerdSim.h
// HerdSim.h : main header file for the HERDSIM application
#ifndef __AFXWIN_H__
#error include 'stdafx.h' before including this file for PCH
#endif
#include "resource.h" // main symbols
////////////////////////////////////////////////////////////
// CHerdSimApp:
// See HerdSim.cpp for the implementation of this class
class CHerdSimApp : public CWinApp
{
public:
    CHerdSimApp();
    // Overrides
    // ClassWizard generated virtual function overrides
    ///{{{AFX_VIRTUAL(CHerdSimApp)
    public:
    virtual BOOL InitInstance();
    ///}}AFX_VIRTUAL
    // Implementation
    ///{{{AFX_MSG(CHerdSimApp)
    afx_msg void OnHelpContents();
    ///}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////
HerdSimDlg.h
// HerdSimDlg.h : header file
>Status::Warning("Not implemented") << std::endl;
// class CHerdSimDlg : public CDialog
{ public:
CHerdSimDlg(CWnd* pParent = NULL);
enum { IDD = IDD_HERDSIM_DIALOG };
protected:
virtual void DoDataExchange(CDataExchange* pDX);
protected:
HICON m_hIcon;
virtual BOOL OnInitDialog();
afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
afx_msg void OnPaint();
afx_msg HCURSOR OnQueryDragIcon();
afx_msg void OnHelpContents();
DECLARE_MESSAGE_MAP()
};

// class CInput2 : public CDialog
{ private:
BOOL startrefresh;
public:
void Startup();
CInput2(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT2 }
CButton m_next;
BOOL m_AI;
BOOL m_black;
BOOL m_buypreg;
int m_conum;
double m_costbull;
int m_cullpol;
BOOL m_implantheifer;
BOOL m_implantsheifer;
BOOL m_implantssteer;
BOOL m_implantsteer;
BOOL m_iono;
int m_numbulls;
double m_salvbull;
int m_yearsbull;
int m_shimplants;
int m_ssimplants;
int m_cullage;
BOOL m_buycalves;
BOOL m_selecwiherd;
int m_repbreed;
int m_reppure;
int m_repsex;
protected:
virtual void DoDataExchange(CDataExchange* pDX);
protected:
afx_msg void OnNext();
afx_msg void OnImplantssteer();
afx_msg void OnImplantsheifer();
afx_msg void OnEarlybs();
afx_msg void OnMenu();
afx_msg void OnPrev();
virtual BOOL OnInitDialog();
afx_msg void OnBuycalves();
afx_msg void OnHelpContents();
DECLARE_MESSAGE_MAP()
};

///////////////////////////////////////////////////////////class CInput1 : public CDialog
{
public:
  CInput1(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT1 };  
CBUTTON m_next;
CString m_address;
CString m_farmman;
CString m_farmname;
CString m_farmowner;
CString m_liveman;
protected:
  virtual void DoDataExchange(CDataExchange* pDX);
protected:
  afx_msg void OnNext();
  afx_msg void OnMenu();
  afx_msg void OnHelpContents();
  DECLARE_MESSAGE_MAP()
};

///////////////////////////////////////////////////////////class CMainMenu : public CDialog
{
public:
  CMainMenu(CWnd* pParent = NULL);
enum { IDD = IDD_MAINMENU };  
CBUTTON m_startsim;
CBUTTON m_quitsim;
CBUTTON m_nutrit;
CBUTTON m_market;
CBUTTON m_forage;
CBUTTON m_cow;
protected:
  virtual void DoDataExchange(CDataExchange* pDX);
protected:
  afx_msg void OnCow();
  afx_msg void OnForage();
  afx_msg void OnMarket();
}
afx_msg void OnNutrit();
afx_msg void OnQuitsim();
afx_msg void OnStartsim();
afx_msg void OnHelpContents();
DECLARE_MESSAGE_MAP()
};

////////////////////////////////////////////////////////////
/CInput2A dialog
class CInput2A : public CDialog
{
public:
CInput2A(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT2A }; double m_angus;
double m_hereford;
int m_bred;
int m_bullbreed;
double m_phereford;
double m_shorthorn;
double m_charol;
double m_chianina;
double m_gelb;
double m_limousin;
double m_main;
double m_pinz;
int m_pure;
double m_simmmental;
double m_tarentaise;
BOOL m_ezcalf;
int m_ezbullbreed;
BOOL m_calfese;
afx_msg void OnBred();
afx_msg void OnEzCalf();
afx_msg void OnPrev();
afx_msg void OnMenu();
virtual BOOL OnInitDialog();
afx_msg void OnNext();
afx_msg void OnHelpContents();
DECLARE_MESSAGE_MAP()
};

////////////////////////////////////////////////////////////
/CInput3 dialog
class CInput3 : public CDialog
{
private:
BOOL startrefresh;
public:
CInput3(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT3 }; CButton m_next;
int m_culled;
double m_diff;
int m_heifer;
int m_sheifer;
int m_ssteer;
int m_steer;
double m_year1;
double m_year2;
double m_year3;
double m_syear1;
double m_syear2;
double m_syear3;
double m_cyear1;
double m_cyear2;
double m_cyear3;
int m_orphan;
int m_hedg1;
int m_hedg2;
int m_onite;
int m_dist;

virtual void DoDataExchange(CDataExchange* pDX);

afx_msg void OnNext();

afx_msg void OnMenu();

afx_msg void Startup();

virtual BOOL OnInitDialog();

afx_msg void OnHedgecalves();

afx_msg void OnMarketingcattle();

//}}AFX_MSG
DECLARE_MESSAGE_MAP()

//=== CInput4 dialog

class CInput4 : public CDialog
{
    // Construction

public:
    CInput4(CWnd* pParent = NULL);

enum { IDD = IDD_INPUT4 };
double m_aicharge;
double m_calftrans;
double m_fence;
double m_implants;
double m_insurance;
double m_intr;
double m_intrs;
double m_labor;
double m_laborcharge;
double m_machine;
double m_maint;
double m_mincosts;
double m_sfence;
double m_slabor;
double m_smachine;
double m_smaint;
double m_smincost;
double m_ssupplies;
double m_strans;
double m_supplies;
double m_sutils;
double m_svet;
double m_taxrate;
double m_trans;
double m_utils;
double m_vet;
virtual void DoDataExchange(CDataExchange* pDX);
afx_msg void OnPrev();
afx_msg void OnMenu();
afx_msg void OnHelpContents();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

/ CInput5 dialog
class CInput5 : public CDialog
{
CInput5(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT5 };
int m_erosion;
int m_graze;
int m_maint;
int m_region;
double m_rentalrate;
double m_respadd;
double m_respmaint;
double m_slope;
int m_terrain;
int m_weed;
double m_sellhay;
double m_haycost;
virtual void DoDataExchange(CDataExchange* pDX);
afx_msg void OnPrev();
afx_msg void OnMenu();
afx_msg void OnHelpContents();
virtual BOOL OnInitDialog();
DECLARE_MESSAGE_MAP()
};

/ CInput6 dialog
class CInput6 : public CDialog
{
// Construction
public:
CInput6(CWnd* pParent = NULL);
};
enum { IDD = IDD_INPUT6 }

double m_alfalfa;
double m_bahia;
double m_bermuda100;
double m_bermuda200;
double m_bermuda250;
double m_bermudar;
double m_dallis;
double m_fescue;
double m_fescuelad;
double m_kentucky;
double m_kentuckyw;
double m_kudza;
double m_lespedza;
double m_orchard;
double m_orchardl;
double m_orchardr;
double m_pearl;
double m_rye;
double m_sericea;
double m_sudan;
double m_switch;
double m_total;

protected:
virtual void DoDataExchange(CDataExchange* pDX);

protected:
afx_msg void OnPrev();
afx_msg void OnMenu();
afx_msg void OnHelpContents();
DECLARE_MESSAGE_MAP()

CInput7 dialog
class CInput7 : public CDialog
{
public:
CInput7(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT7 }

double m_grcost1;
double m_grcost10;
double m_grcost11;
double m_grcost12;
double m_grcost13;
double m_grcost14;
double m_grcost15;
double m_grcost16;
double m_grcost17;
double m_grcost18;
double m_grcost19;
double m_grcost2;
double m_grcost20;
double m_grcost3;
double m_grcost4;
double m_grcost5;
double m_grcost6;
double m_grcost7;
double m_grcost8;
double m_grcost9;
CString m_sopt1;
CString m_sopt2;
CString m_sopt3;
CString m_sopt4;
CString m_sopt5;
double m_su1;
double m_su10;
double m_su11;
double m_su12;
double m_su13;
double m_su14;
double m_su15;
double m_su16;
double m_su17;
double m_su18;
double m_su19;
double m_su2;
double m_su20;
double m_su3;
double m_su4;
double m_su5;
double m_su6;
double m_su7;
double m_su8;
double m_su9;
double m_total;
BOOL m_usesupp;
protected:
virtual void DoDataExchange(CDataExchange* pDX);
protected:
afx_msg void OnMenu();
afx_msg void OnNext();
afx_msg void OnUsesupp();
virtual BOOL OnInitDialog();
afx_msg void OnHelpContents();
DECLARE_MESSAGE_MAP();

////////////////////////////////////////////////////////////
/ CInput8 dialog
class CInput8 : public CDialog {
public:
CInput8(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT8 };
double  m_sdm16;
double  m_sdm17;
double  m_sdm18;
double  m_sdm19;
double  m_sdm20;
double  m_sme16;
double  m_sme17;
double  m_sme18;
double  m_sme19;
double  m_sme20;
double  m_sneg16;
double  m_sneg17;
double  m_sneg18;
double  m_sneg19;
double  m_sneg20;
double  m_snem16;
double  m_snem17;
double  m_snem18;
double  m_snem19;
double  m_snem20;
CString  m_sopt1;
CString  m_sopt2;
CString  m_sopt3;
CString  m_sopt4;
CString  m_sopt5;
double  m_stdn16;
double  m_stdn17;
double  m_stdn18;
double  m_stdn19;
double  m_stdn20;
protected:
virtual void DoDataExchange(CDataExchange* pDX);
protected:
afx_msg void OnMain();
afx_msg void OnPrev();
afx_msg void OnHelpContents();
DECLARE_MESSAGE_MAP()
};

////////////////////////////////////////////////////////////
/CInput2B dialog
class CInput2B : public CDialog
{
   // Construction
public:
CInput2B(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT2B };
int    m_fallday;
int    m_falllength;
int    m_fallmonth;
BOOL   m_shbse;
BOOL   m_shbse2;
int m_shbse2day;
int m_shbse2month;
int m_shbseyday;
int m_shbsemonth;
int m_breedingseasons;
int m_springday;
int m_springlength;
int m_springmonth;
int m_weansday;
int m_weansmonth;
int m_weanfday;
int m_weanfmonth;
protected:
virtual void DoDataExchange(CDataExchange* pDX);     pro-
tected:
afx_msg void OnBack();
afx_msg void OnPrev();
afx_msg void OnMenu();
afx_msg void OnCalving();
afx_msg void OnSpringcalving();
afx_msg void OnShbse2();
afx_msg void OnShbse();
virtual BOOL OnInitDialog();
afx_msg void OnHelpContents();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

 YearOption dialog
class YearOption : public CDialog
{
public:
    YearOption(CWnd* pParent = NULL);
enum { IDD = IDD_YEAROPTION };            BOOL m_changesire;
    bool m_changesire;
    int m_sire;
    int m_sire2;
    CStrint religion m_caption;
protected:
    virtual void DoDataExchange(CDataExchange* pDX);
    protected:
    afx_msg void OnChangesire();
    virtual BOOL OnInitDialog();
    afx_msg void OnHelpContents();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

CDieHerd dialog
class CDieHerd : public CDialog
{ public:
CDieHerd(CWnd* pParent = NULL);
enum { IDD = IDD_HERDDIE }; CString m_prob;
protected:
virtual void DoDataExchange(CDataExchange* pDX);
protected:
DECLARE_MESSAGE_MAP()
};

class CINPUT9 : public CDialog
{
// Construction
public:
void StartUp();
CINPUT9(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT9 }
int m_dayin1;
int m_dayin2;
int m_dayin3;
int m_dayin4;
int m_dayin5;
int m_dayin6;
int m_dayout1;
int m_dayout2;
int m_dayout3;
int m_dayout4;
int m_dayout5;
int m_dayout6;
int m_hland1;
int m_hland2;
int m_hland3;
int m_hland4;
int m_hland5;
int m_hland6;
in
m_monthin1;
in
m_monthin2;
in
m_monthin3;
in
m_monthin4;
in
m_monthin5;
in
m_monthin6;
in
m_monthout1;
in
m_monthout2;
in
m_monthout3;
in
m_monthout4;
in
m_monthout5;
in
m_monthout6;
in
m_tpas1;
in
m_tpas2;

959
int m_tpas3;
int m_tpas4;
int m_tpas5;
int m_tpas6;
int m_wland1;
int m_wland2;
int m_wland3;
int m_wland4;
int m_wland5;
int m_wland6;
int m_numpas;
int m_numpasture;
protected:
virtual void DoDataExchange(CDataExchange* pDX);
protected:
virtual BOOL OnInitDialog();
afx_msg void OnNext();
afx_msg void OnChangeNumpasture();
DECLARE_MESSAGE_MAP();
};

////////////////////////////////////////////////////////////
/CInput10 dialog
class CInput10 : public CDialog
{
// Construction
public:
void StartUp();
CInput10(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT10 };
```cpp
int m_monthrin5;
int m_monthrin6;
int m_monthrout1;
int m_monthrout2;
int m_monthrout3;
int m_monthrout4;
int m_monthrout5;
int m_monthrout6;
int m_numrpasture;
int m_trpas1;
int m_trpas2;
int m_trpas3;
int m_trpas4;
int m_trpas5;
int m_trpas6;
int m_wrland1;
int m_wrland2;
int m_wrland3;
int m_wrland4;
int m_wrland5;
int m_wrland6;
protected:
virtual void DoDataExchange(CDataExchange* pDX);    //
// DDX/DDV support
protected:
virtual BOOL OnInitDialog();
afx_msg void OnPrev();
afx_msg void OnNext();
afx_msg void OnChangeNumrpasture();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
CProgress(CWnd* pParent = NULL);
enum { IDD = IDD_PROGRESS }
CString m_caption1;
protected:
virtual void DoDataExchange(CDataExchange* pDX);
protected:
DECLARE_MESSAGE_MAP()
};
class CProgress : public CDialog
{
public:
void onStartUp(int numOfYears);
void onStepFunction();
CProgress(CWnd* pParent = NULL);
enum { IDD = IDD_PROGRESS }
CString m_caption1;
protected:
virtual void DoDataExchange(CDataExchange* pDX);
protected:
DECLARE_MESSAGE_MAP()
};
Cinput11 dialog
class Cinput11 : public CDialog
{
// Construction
```
public:
Cinput11(CWnd* pParent = NULL);
enum { IDD = IDD_INPUT11 };  
BOOL m_adg1;
BOOL m_seasonal;
double m_antiadg;
double m_antiadg1;
double m_antiadg2;
double m_antiadg3;
double m_antiadg4;
int m_retstocker;
protected:
virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
protected:
afx_msg void OnPrev();
afx_msg void OnNext();
afx_msg void OnMenu();
afx_msg void OnHelp();
afx_msg void OnAdg();
virtual BOOL OnInitDialog();
DECLARE_MESSAGE_MAP();
}
Main.h
#ifndef MAIN_H
#define MAIN_H
class Main{
public:
Main();
void Start();
};
#endif
#ifndef MARKET_H
#define MARKET_H
#include "cow.h"
#include <iostream.h>
#include <fstream.h>
class Market{
private:
    double AICS;
    double BASS;
    double BFRC;
    double BFRS;
    double BSAL;
    double DMandSC;
    double DPREM;
    double EPREM;
    double FICA;
    double HDEX;
    double HEDG1;
    double HEDG2;
    double HOUR;
    double HOURS;
    double INSURS;
    double INTR;
    double INTRS;
    double IPCS;
    double LORT;
    double LORTS;
    double LSCS;
    double LSCSS;
    double MCCO;
    double MCCOS;
    double PMNA;
    double PMNS;
    double PRSA1;
    double PRSA2;
    double PRSA3;
    double SPRSA1;
    double SPRSA2;
    double SPRSA3;
    double CPRSA1;
    double CPRSA2;
    double CPRSA3;
    double SALT;
    double SMCS;
    double SMCSS;
    double TCCS;
    double TCSC;
    double TCSCS;
    double UTLC;
    double UTLCH;
    double VMCS;
}
double VMCSS;
double WAGE;
int DIST;
int NBUL;
int ONIT;
int SALC;
int SALH;
int SALO;
int SALS;
int SALSH;
int SALSS;
double CCRV;
double PRIS;
double TCCR;
double NCCS;
double TWCC;
double TBDCC;
double TDSPPC;
double TSHKC;
double ASHKC;
double YDCAC;
double TFEAC;
double OCRV;
double TOCR;
double ASHKOC;
double TSHKOC;
double TDSPOC;
double TBDOC;
double THFR;
int NOC;
int NHFC;
int NSTC;
int TWOS;
double TWSS;
double TWHS;
double TBDH;
double TDSPPH;
double TDSPPS;
double TESPH;
double TESPS;
double TFECA;
double TSHKH;
double TSHKS;
double TSTR;
double TBDS;
double ASHKH;
double ASHK;
double YDCA;
double HFRV;
double STRV;
int NSHFC;
int NSSTC;
double TWSHS;
double TWSSS;
double TBDSSH;
double TBDDS;
double TDSPPSH;
double TDSPSS;
double TESPSS;
double TESPSSH;
double TFECAS;
double TSHFR;
double TSHKSH;
double TSHKSS;
double TSSTR;
double ASHKSH;
double ASHKSS;
double YDCAS;
double TotalPurchaseCost;
private:
void UseInput3();
void UseInput4();
public:
Market();
void Init();
int SellCullCows(AnimalRecord *);
int SellOrphans(AnimalRecord *);
void SellWeanlings(AnimalRecord *);
int SellStockerCalves(AnimalRecord *);
void BudgetReport(ofstream&);
void PurchaseLivestock(AnimalRecord *);
void ResetMarketVariables();
};
#endif MARKET_H
Nutrit.h
#ifndef NUTRIT_H
#define NUTRIT_H
#include "cow.h"

class Nutrition{
  double GRCost[21];
  double SDM[21];
  double SME[21];
  double SNEG[21];
  double SNEM[21];
  CString SOPT;
  CString SOPT1;
  CString SOPT2;
  CString SOPT3;
  CString SOPT4;
  CString SOPT5;
  double STDN[21];
  double SU[21];
  double APDMIF;
  double ASPDMIF;
  double PDMI;
  double SAME;
  double SANEG;
  double SANEM;
  double GRAZE;
  double GUNIT;
  double SUPP;
private:
  void UseInput7();
  void UseInput8();
public:
  Nutrition();
  void Init();
  void MainNutrition(AnimalRecord *);
  void FeedUsage();
  void DWGAIN(AnimalRecord *);
  void ResetNutritionVariables();
};
#endif
StdAfx.h
//stdafx.h : include file for standard system
//include files, or project specific include
//files that are used frequently, but
//are changed infrequently
//
#define VC_EXTRALEAN
#include <afxwin.h>
#include <afxext.h>
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>
#endif
Tools.h
#ifndef TOOLS_H
#define TOOLS_H

int Julian(int , int );
double GetStochasticNumber(double , double );
void cleardevice();
int GetRandom();
#endif
Vita

James Henry Schick was born at Mount Vernon, Illinois. He received his Bachelor of Science in Animal Science in June 1967 and then completed one year towards a Master of Business Administration from the University of Illinois, Champaign, Illinois before withdrawing for military service. In 1971, he went to work for Western Plains Drilling Company, Midland, Texas selling oil and gas investments. He later worked as a commodity broker for duPont Walston, Inc. In 1975, he joined Dowell Schlumberger Int., Ltd. where he held various international management and staff positions. In 1981, he formed Schick Drilling Co., Inc., a oil and gas well contract drilling firm. In 1993, he entered graduate school at West Texas A&M University where he completed a Master of Agriculture in Agricultural Business and Economics in August 1994. He entered Virginia Tech in August 1994.