```
1  ************************************************************
2  *   PROGRAMMED BY : Andrew Gharios
3  *   STUDENT ID    : 1449366
4  *   CLASS         : M-Th 5-7:20p
5  *   LAB #12       : Intro to OOP
6  ************************************************************
7  USING ARRAY METHOD.
8  ***********************************
9  * WELCOME TO THE SHEEP LIST MANAGER *
10 ***********************************
11
12 SHEEP LIST MANAGER
13 1 - Add Sheep
14 2 - Output 1st Sheep
15 3 - Find Sheep
16 4 - List Size
17 5 - Output List
18 6 - Clear List
19 0 - Exit
20 Enter a command: 1
21
22 Sheep Name: Fluffy
23 Sheep Age:  1
24
25 The sheep...
26 Sheep Name: Fluffy
27 Sheep Age:  1
28 Has been added!
29
30 Enter a command: 2
31
32 NAME            AGE
33 -------------- ---
34 Fluffy          1
35
36 Is at the front of the list.
37
38 Enter a command: 1
39
40 Sheep Name: Maa
41 Sheep Age:  3
42
43 The sheep...
44 Sheep Name: Maa
45 Sheep Age:  3
46 Has been added!
47
48 Enter a command: 4
49 There are 2 sheep in the list.
```

```
50
51  Enter a command: 5
52
53
54  NAME          AGE
55  ------------- ---
56  Fluffy          1
57  Maa             3
58  There are 2 sheep in the list.
59
60  Enter a command: 1
61
62  Sheep Name: Baa Baa
63  Sheep Age:  2
64
65  The sheep...
66  Sheep Name: Baa Baa
67  Sheep Age:  2
68  Has been added!
69
70  Enter a command: 5
71
72
73  NAME          AGE
74  ------------- ---
75  Fluffy          1
76  Maa             3
77  Baa Baa         2
78  There are 3 sheep in the list.
79
80  Enter a command: 4
81  There are 3 sheep in the list.
82
83  Enter a command: 3
84
85  Who are you looking for? Baa Baa
86
87
88  NAME          AGE
89  ------------- ---
90  Baa Baa         2
91  Has been found.
92
93  Enter a command: 6
94  The list has been cleared!
95
96  Enter a command: 6
97  The list is empty!
98
```

```
 99  Enter a command: 5
100
101  Can't display an empty list
102  Nobody is in the list!
103
104  Enter a command: 4
105  Nobody is in the list!
106
107  Enter a command: 3
108
109  Who are you looking for? Baa
110
111  Can't search an empty list
112
113  Enter a command: 2
114  The list is empty!
115
116  Enter a command: 7
117
118  **** The number 7 is an invalid entry     *****
119  **** Please input a number between 0 and 6 *****
120
121  Enter a command: 0
```

```
1  #ifndef HEADER_H_
2  #define HEADER_H_
3
4  #include <iostream> // cin, cout.
5  #include <string>   // string datatype variables.
6  #include <fstream>  // Fstream files.
7  #include <iomanip>  // fixed, setw, setprecision.
8  #include <ostream>  // Ostream data type.
9  #include <ctype.h>
10
11 using namespace std;
12
13 const int NAME_SIZE = 15; // SETW size for Name column.
14
15 /*****************************************************************************
16  * PrintHeaderFile
17  *   This function will output the header information
18  *
19                                                                              ↵
   *****************************************************************************/
20 void PrintHeaderFile(ostream& output,      // IN  - output datatype.
21     string asName,        // IN  - assignment name
22     int asNum,             // IN  - assignment number
23     string studentName,   // IN  - student's name
24     string classInfo,     // IN  - class that is being taken
25     char asType,          // IN  - assignment type
26     long long studentID); // IN  - student ID
27
28
29 #endif
30
```

```
/*****************************************************************************
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* LAB #13    :
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 7/27/21
*****************************************************************************/
#include "Header.h"
#include "Sheep.h"
#include "Array.h"
#include "List.h"

/*****************************************************************************
* Arrays & Linked Lists of Sheep
*----------------------------------------------------------------------------
* This program is a sheep list manager. It will prompt the user with a menu to
* add sheep to the list, output the first sheep, display the list, find a
* specific sheep, find the list size or clear the list.
*----------------------------------------------------------------------------
* INPUT:
*  input : Menu selection.
*  sheepName : Name for sheep to be added.
*  sheepAge  : Age for sheep to be added.
*****************************************************************************/
int main()
{
    /*************************************************************************
    * CONSTANTS
    *------------------------------------------------------------------------
    * OUTPUT - USED FOR CLASS HEADING
    *------------------------------------------------------------------------
    * PROGRAMMER : Programmer's Name
    * CLASS   : Student's Course
    * SECTION    : Class Days and Times
```

```
36      * LAB_NUM    : Lab Number (specific to this lab)
37      * LAB_NAME   : Title of the Lab
38      ********************************************************************** ↵
          /
39
40      const string AS_NAME = "Intro to OOP";
41      const int AS_NUM = 12;
42      const string STUDENT_NAME = "Andrew Gharios";
43      const string CLASS_INFO = "M-Th 5-7:20p";
44      const char AS_TYPE = 'L';
45      const long long STUDENT_ID = 1449366;
46
47      Array  farm;  // CALC - Farm class using Array implementation.
48    //Array  farm;      // CALC - Farm class using Linked-list implementation.
49      Sheep  sheep;     // CALC      - Sheep class object.
50      int    input;     // IN & CALC - Menu selection.
51      bool   invalid;   // CALC      - Input validation.
52      string sheepName; // IN & CALC - Sheep name input.
53      int    sheepAge;  // IN & CALC - Sheep age input.
54
55      invalid = false;
56
57
58      PrintHeaderFile(cout, AS_NAME, AS_NUM, STUDENT_NAME, CLASS_INFO,
59          AS_TYPE, STUDENT_ID);
60
61      cout << "\n************************************\n";
62      cout << "* WELCOME TO THE SHEEP LIST MANAGER *\n";
63      cout << "************************************\n";
64
65      cout << "\nSHEEP LIST MANAGER\n";
66          cout << "1 - Add Sheep\n";
67      cout << "2 - Output 1st Sheep\n";
68      cout << "3 - Find Sheep\n";
69      cout << "4 - List Size\n";
70      cout << "5 - Output List\n";
71      cout << "6 - Clear List\n";
72      cout << "0 - Exit";
73
74      do
75      {
76          do
77          {
78              invalid = false;
79              cout << "\nEnter a command: ";
80              if (!(cin >> input))
81              {
82                  cout << "\n**** Please enter a NUMBER between 0 and 6 ****\n";
83                  cin.clear();
```

```
84                  cin.ignore(numeric_limits<streamsize>::max(), '\n');
85                  invalid = true;
86              }
87              else if (input < 0 || input > 6)
88              {
89
90                  cout << "\n**** The number " << input << " is an invalid entry ↵
                        *****\n";
91                  cout << "**** Please input a number between 0 and 6 *****\n";
92                  invalid = true;
93              }
94
95
96          } while (invalid);
97
98          cin.ignore(numeric_limits<streamsize>::max(), '\n');
99
100         switch (input)
101         {
102         case 1:
103             cout << "\nSheep Name: ";
104             getline(cin, sheepName);
105             cout << "Sheep Age:  ";
106             cin >> sheepAge;
107             cin.ignore(10000, '\n');
108
109             sheep.SetInitialValues(sheepName, sheepAge);
110             farm.AddSheep(sheep);
111             break;
112         case 2:
113
114             farm.OutputFirst();
115             break;
116         case 3:
117             cout << "\nWho are you looking for? ";
118             getline(cin, sheepName);
119
120             farm.FindSheep(sheepName);
121             break;
122         case 4:
123             farm.ListSize();
124             break;
125         case 5:
126             farm.DisplayList();
127             farm.ListSize();
128             break;
129         case 6:
130             farm.ClearList();
131             break;
```

```
132           }
133       } while (input != 0);
134
135   }
```

```cpp
#include "Header.h"

/*****************************************************************************
 * PrintHeaderFile
 *   This function will output the header information
 *
 * PRE-CONDITIONS
 *   The following parameters need to have a defined value prior to calling
 *   the function
 *          asName: The name of the assignment given in the course
 *          asNum: The number of the assignment given in the course
 *          studentName: The name of the student writing the code
 *          classInfo: The course name, date, and time of the class
 *          asType: Will either output as a lab or an assignment
 *          studentID: The Identification Number of the student
 *
*****************************************************************************/

void PrintHeaderFile(ostream& output,       // IN - output datatype.
    string asName,      // IN - assignment name
    int asNum,          // IN - assignment number
    string studentName,  // IN - student's name
    string classInfo,    // IN - class that is being taken
    char asType,         // IN - assignment type
    long long studentID) // IN - student ID
{
    output << left;
    output << "*************************************************************   \n";
    output << "*   PROGRAMMED BY : " << studentName << endl;
    output << "*   " << setw(14) << "STUDENT ID " << ": " << studentID << endl;
    output << "*   " << setw(14) << "CLASS " << ": " << classInfo << endl;
    output << "*   ";

    // PROCESSING - This will adjust setws and format appropriately based
    //              on if this is a lab 'L' or assignment

    if (toupper(asType) == 'L')
    {
        output << "LAB #" << setw(9);
    }
    else
    {
        output << "ASSIGNMENT #" << setw(2);
    }
    output << asNum << ": " << asName << endl;
    output << "*************************************************************";
    output << right << endl;
```

```
47
48      return;
49  }
```

```
1  #ifndef ARRAY_H_
2  #define ARRAY_H_
3
4  #include "Sheep.h"
5  #include "Header.h"
6
7  const int AR_SIZE = 50; // Array-size
8
9  class Array
10 {
11 public:
12
13     Array();  // CONSTRUCTOR.
14     ~Array(); // DESCONSTRUCTOR.
15
16     /*********************
17      **    MUTATORS    **
18      *********************/
19     void AddSheep(Sheep sheep);        // Adds a sheep to the list.
20     void ClearList();                  // Clears the list from all sheep.
21
22     /*********************
23      **    ACCESSORS    **
24      *********************/
25     void DisplayList() const;          // Displays the entire list.
26     void OutputFirst() const;          // Outputs the first sheep in the list.
27     void FindSheep(string name) const; // Searches the list for the inputted
         name.
28     void ListSize() const;             // Displays the size of the list.
29
30 private:
31
32     /*********************
33      **   ATTRIBUTES    **
34      *********************/
35     int sheepCount;        // CALC - Sheep count.
36     Sheep farmAr[AR_SIZE]; // CALC - Farm array to hold all sheep.
37 };
38
39 #endif
```

```cpp
#include "Header.h"
#include "Sheep.h"
#include "Array.h"

Array::Array() // CONSTRUCTOR.
{
    sheepCount = 0;
}

Array::~Array(){} // DESCONSTRUCTOR.

/*********************
**    MUTATORS    **
*********************/
void Array::AddSheep(Sheep sheep)
{
    int sheepAge;     // CALC - Sheepage temp storage.
    string sheepName; // CALC - Sheepname temp storage.

    if (sheepCount < AR_SIZE)
    {
        farmAr[sheepCount] = sheep;
        farmAr[sheepCount].GetValues(sheepName, sheepAge);
        cout << "\nThe sheep...\n";
        cout << "Sheep Name: " << sheepName;
        cout << "\nSheep Age:  " << sheepAge;
        cout << "\nHas been added!\n";
        sheepCount++;
    }
    else
    {
        cout << "Out of memory sheep wasn't added\n";
    }
}

void Array::ClearList()
{
    if (sheepCount > 0)
    {
        while (sheepCount > 0)
        {
            sheepCount--;
        }
        cout << "The list has been cleared!" << endl;
    }
    else
    {
        cout << "The list is empty!" << endl;
    }
```

```
50  }
51
52
53  /*********************
54  **    ACCESSORS    **
55  *********************/
56  void Array::DisplayList() const
57  {
58      int counter;  // CALC - counter to manipulate Array index.
59      int sheepAge;     // CALC - Sheepage temp storage.
60      string sheepName; // CALC - Sheepname temp storage.
61
62      counter = 0;
63
64      if (sheepCount == 0)
65      {
66          cout << "\nCan\'t display an empty list\n";
67      }
68      else
69      {
70          cout << endl;
71          cout << left;
72          cout << setw(NAME_SIZE) << "\nNAME";
73          cout << "AGE\n";
74          cout << setw(NAME_SIZE) << string(NAME_SIZE - 1, '-');
75          cout << string(3, '-') << endl;
76          cout << right;
77
78          while (counter < sheepCount)
79          {
80              cout << left;
81              farmAr[counter].GetValues(sheepName, sheepAge);
82              cout << setw(NAME_SIZE) << sheepName;
83              cout << " " << sheepAge;
84              cout << endl;
85              cout << right;
86
87              counter++;
88          }
89      }
90  }
91
92  void Array::OutputFirst() const
93  {
94      int sheepAge;     // CALC - Sheepage temp storage.
95      string sheepName; // CALC - Sheepname temp storage.
96
97      if (sheepCount > 0)
98      {
```

```
 99          farmAr[0].GetValues(sheepName, sheepAge);
100          cout << endl;
101          cout << left;
102          cout << setw(NAME_SIZE) << "NAME";
103          cout << " AGE\n";
104          cout << setw(NAME_SIZE) << string(NAME_SIZE - 1, '-');
105          cout << string(3, '-') << endl;
106          cout << setw(NAME_SIZE) << sheepName;
107          cout << " " << sheepAge;
108          cout << "\n\nIs at the front of the list.\n";
109          cout << right;
110      }
111      else
112      {
113          cout << "The list is empty!\n";
114      }
115  }
116
117  void Array::FindSheep(string name) const
118  {
119      int counter;  // CALC - counter to manipulate Array index.
120      bool found;       // CALC - If sheep was found or not.
121      int sheepAge;     // CALC - Sheepage temp storage.
122      string sheepName; // CALC - Sheepname temp storage.
123
124      found = false;
125      counter = 0;
126
127      if (sheepCount > 0)
128      {
129          while (counter < sheepCount && !found)
130          {
131              if (farmAr[counter].GetName() == name)
132              {
133                  found = true;
134                  farmAr[counter].GetValues(sheepName, sheepAge);
135                  cout << endl;
136                  cout << left;
137                  cout << setw(NAME_SIZE) << "\nNAME";
138                  cout << "AGE\n";
139                  cout << setw(NAME_SIZE) << string(NAME_SIZE - 1, '-');
140                  cout << string(3, '-') << endl;;
141                  cout << setw(NAME_SIZE) << sheepName;
142                  cout << " " << sheepAge; // age output
143                  cout << endl;
144                  cout << "Has been found.\n";
145                  cout << right;
146              }
147              counter++;
```

```
148         }
149
150         if (!found)
151         {
152             cout << "I\'m sorry, \"" << name << "\" was NOT found!\n";
153         }
154     }
155     else
156     {
157         cout << "\nCan't search an empty list\n";
158     }
159 }
160
161
162 void Array::ListSize() const
163 {
164     if (sheepCount == 0)
165     {
166         cout << "Nobody is in the list!\n";
167     }
168     else if (sheepCount == 1)
169     {
170         cout << "There is one sheep in the list.\n";
171     }
172     else
173     {
174         cout << "There are " << sheepCount << " sheep in the list.\n";
175     }
176 }
177
178 /*****************************************************************************
    **
179 * AddSheep
180 *   This function will add a new sheep to the tail of the linked list.
181 * INPUTS:
182 *   sheep : sheep object.
183 *
184 * No outputs.
185 *****************************************************************************
    */
186
187 /*****************************************************************************
    **
188 * ClearList
189 *   This function will clear the linked list of all sheep.
190 * No inputs.
```

```
191  *
192  * No outputs.
193  *                                                                        ↵
       ************************************************************************ ↵
       */
194
195  /                                                                        ↵
       ************************************************************************ ↵
       **
196  * DisplayList
197  *   This function will display the entire linked list in a specific format.
198  * No inputs.
199  *
200  * No outputs.
201  *                                                                        ↵
       ************************************************************************ ↵
       */
202
203  /                                                                        ↵
       ************************************************************************ ↵
       **
204  * OutputFirst
205  *   This function will display the information of the first sheep in the   ↵
       linked
206  *   list.
207  * No inputs.
208  *
209  * No outputs.
210  *                                                                        ↵
       ************************************************************************ ↵
       */
211
212  /                                                                        ↵
       ************************************************************************ ↵
       **
213  * FindSheep
214  *   This function will look for the inputed sheep name inside the linked list
215  *   and output the sheep's information if it was found.
216  *
217  * INPUTS:
218  *  name : Sheep's name to search for.
219  *
220  * No outputs.
221  *                                                                        ↵
       ************************************************************************ ↵
       */
222
223  /                                                                        ↵
       ************************************************************************ ↵
```

```
      **
224   * ListSize
225   *   This function will display the size of the linked-list.
226   * No inputs.
227   *
228   * No outputs.
229   *                                                                      ↵
      ************************************************************************ ↵
      */
```

```
1  #ifndef ANIMAL_H_
2  #define ANIMAL_H_
3
4  #include "Header.h"
5
6  class Sheep
7  {
8  public:
9      Sheep();  // CONSTRUCTOR.
10     ~Sheep(); // DESCONTRUCTOR.
11
12     /*********************
13      **    MUTATORS    **
14      *********************/
15
16     void SetInitialValues(string aName,     // sets the initial values for the
         sheep.
17                           int aAge);
18
19     /*********************
20      **    ACCESSORS    **
21      *********************/
22
23     void GetValues(string &sheepName,
24                    int    &sheepAge) const; // returns values of the sheep.
25     string GetName() const;                 // returns the name of the sheep.
26
27 private:
28     string   name;  // IN & OUT - Animal name.
29     int       age;  // IN & OUT - Animal age.
30 };
31
32 #endif
33
34
35
```

```cpp
#include "Header.h"
#include "Sheep.h"

Sheep::Sheep() // CONSTRUCTOR.
{
    age = 0;
}

Sheep ::~Sheep() {} // DECONSTRUCTOR.

/*********************
**    MUTATORS    **
*********************/
void Sheep::SetInitialValues(string aName,
    int aAge)
{
    name = aName;
    age = aAge;
}


/*********************
**    ACCESSORS    **
*********************/
void Sheep::GetValues(string &sheepName,
                      int    &sheepAge) const
{
    sheepName = name;
    sheepAge  = age;
}

string Sheep::GetName() const
{
    return name;
}

/*****************************************************************************
* SetInitialValues
*   This function will set all the initial values for the sheep object.
*
* INPUTS:
*   aName  : choice of name.
*   aAge   : choice of age.
*
* No outputs.
*                                                                            ↵
  ***************************************************************************↵
  /

```

```
48  /*****************************************************************************
49  * GetValues
50  *   This function will return the values of the sheep.
51  *
52  * No inputs.
53  *
54  *  OUTPUTS:
55  *   name : sheep's name.
56  *   age  : sheep's age.
57  *                                                                            ↵
      ***************************************************************************** ↵
      /
58
59  /*****************************************************************************
60  * GetName
61  *   This function will return the name of the sheep.
62  *
63  * No inputs.
64  *
65  *  OUTPUTS:
66  *   name : sheep's name.
67  *                                                                            ↵
      ***************************************************************************** ↵
      /
```

```
1  ************************************************************
2  *   PROGRAMMED BY : Andrew Gharios
3  *   STUDENT ID    : 1449366
4  *   CLASS         : M-Th 5-7:20p
5  *   LAB #12       : Intro to OOP
6  ************************************************************
7  USING LINKED LIST METHOD.
8
9  ***********************************
10 * WELCOME TO THE SHEEP LIST MANAGER *
11 ***********************************
12
13 SHEEP LIST MANAGER
14 1 - Add Sheep
15 2 - Output 1st Sheep
16 3 - Find Sheep
17 4 - List Size
18 5 - Output List
19 6 - Clear List
20 0 - Exit
21 Enter a command: 1
22
23 Sheep Name: Fluffy
24 Sheep Age:  1
25
26 The sheep...
27 Sheep Name: Fluffy
28 Sheep Age: 1
29 Has been added!
30
31 Enter a command: 2
32
33 NAME            AGE
34 -------------- ---
35 Fluffy          1
36
37 Is at the front of the list.
38
39 Enter a command: 1
40
41 Sheep Name: Maa
42 Sheep Age:  3
43
44 The sheep...
45 Sheep Name: Maa
46 Sheep Age: 3
47 Has been added!
48
49 Enter a command: 4
```

```
There are 2 sheep in the list.

Enter a command: 5


NAME          AGE
------------- ---
Fluffy          1
Maa             3
There are 2 sheep in the list.

Enter a command: 1

Sheep Name: Baa Baa
Sheep Age:  2

The sheep...
Sheep Name: Baa Baa
Sheep Age: 2
Has been added!

Enter a command: 5


NAME          AGE
------------- ---
Fluffy          1
Maa             3
Baa Baa         2
There are 3 sheep in the list.

Enter a command: 4
There are 3 sheep in the list.

Enter a command: 3

Who are you looking for? Baa Baa


NAME          AGE
------------- ---
Baa Baa         2
Has been found.

Enter a command: 6
The list has been cleared!

Enter a command: 6
The List is empty!
```

```
 99
100  Enter a command: 5
101
102  Can't display an empty list
103  Nobody is in the list!
104
105  Enter a command: 4
106  Nobody is in the list!
107
108  Enter a command: 3
109
110  Who are you looking for? Baa Baa
111
112  Can't search an empty list
113
114  Enter a command: 2
115  Nobody in FRONT, the list is empty!
116
117  Enter a command: 7
118
119  **** The number 7 is an invalid entry     *****
120  **** Please input a number between 0 and 6 *****
121
122  Enter a command: 0
```

```
1  #ifndef LIST_H_
2  #define LIST_H_
3
4  #include "Sheep.h"
5  #include "Header.h"
6
7  class List
8  {
9      public:
10
11         List();  // CONSTRUCTOR
12         ~List(); // DECONSTRUCTOR.
13
14         /*********************
15          **    MUTATORS    **
16          *********************/
17         void AddSheep(Sheep sheep);        // Adds a sheep to the list.
18         void ClearList();                  // Clears the list from all sheep.
19
20         /*********************
21          **    ACCESSORS    **
22          *********************/
23         void DisplayList() const;          // Displays the entire list.
24         void OutputFirst() const;          // Outputs the first sheep in the
             list.
25         void FindSheep(string name) const; // Searches the list for the
             inputted name.
26         void ListSize() const;             // Displays the size of the list.
27
28     private:
29
30         /*********************
31          **   ATTRIBUTES    **
32          *********************/
33         struct SheepNode
34         {
35             string name; // IN & OUT - Sheep name.
36             int    age;      // IN & OUT - Sheep age.
37             SheepNode* next; // CALC - Next Node
38         };
39
40         int sheepCount;      // CALC     - Sheep count.
41         SheepNode* head; // CALC - Sheep list head pointer.
42 };
43
44 #endif
```

```cpp
#include "Header.h"
#include "List.h"
#include "Sheep.h"

// LINKED LIST METHOD

List::List() // CONSTRUCTOR
{
    head = NULL;
    sheepCount = 0;
}

List::~List() // DECONSTRUCTOR
{
    SheepNode* sheepPtr; // CALC- Sheep pointer.

    sheepPtr = head;

    while (sheepPtr != NULL)
    {
        head = head->next;
        delete sheepPtr;

        sheepPtr = head;
    }
}


/*********************
**    MUTATORS    **
*********************/
void List::AddSheep(Sheep sheep)
{
    SheepNode* ptr;  // CALC- Sheep pointer.
    SheepNode* tail; // CALC- Linked list tail pointer.

    ptr = new SheepNode;
    sheep.GetValues(ptr->name, ptr->age);
    ptr->next = NULL;

    if (ptr != NULL)
    {

        ptr->next = NULL;
        if (head != NULL)
        {
            tail = head;
            while (tail->next != NULL)
            {
```

```
50                  tail = tail->next;
51              }
52              ptr->next  = tail->next;
53              tail->next = ptr;
54          }
55          else
56          {
57              ptr->next = head;
58              head = ptr;
59          }
60
61          cout << "\nThe sheep...\n";
62          cout << "Sheep Name: " << ptr->name;
63          cout << "\nSheep Age: " << ptr->age;
64          cout << "\nHas been added!\n";
65      }
66      else
67      {
68          cout << "Out of memory sheep wasn't added\n";
69      }
70      ptr = NULL;
71      sheepCount++;
72  }
73
74
75  void List::ClearList()
76  {
77      SheepNode* ptr; // CALC - Sheep pointer.
78
79      ptr = head;
80
81      if (head == NULL)
82      {
83          cout << "The List is empty!" << endl;
84      }
85      else
86      {
87
88          while (ptr != NULL)
89          {
90              head = ptr->next;
91              ptr = ptr->next;
92          }
93          cout << "The list has been cleared!" << endl;
94      }
95      ptr = NULL;
96  }
97
98  /*********************
```

```
 99  **    ACCESSORS    **
100  ********************/
101
102 void List::DisplayList() const
103 {
104     SheepNode* perPtr; // CALC  - Sheep pointer.
105
106     perPtr = head;
107
108
109     if (perPtr == NULL)
110     {
111         cout << "\nCan\'t display an empty list\n";
112     }
113     else
114     {
115         cout << endl;
116         cout << left;
117         cout << setw(NAME_SIZE) << "\nNAME";
118         cout << "AGE\n";
119         cout << setw(NAME_SIZE) << string(NAME_SIZE - 1, '-');
120         cout << string(3, '-') << endl;
121         cout << right;
122
123         while (perPtr != NULL)
124         {
125             cout << left;
126             cout << setw(NAME_SIZE) << perPtr->name;
127             cout << " " << perPtr->age;
128             cout << endl;
129             cout << right;
130
131             perPtr = perPtr->next;
132         }
133     }
134
135
136 }
137
138 void List::OutputFirst() const
139 {
140     SheepNode* perPtr; // CALC  - Sheep pointer.
141
142     perPtr = head;
143
144     if (perPtr == NULL)
145     {
146         cout << "Nobody in FRONT, the list is empty!\n";
147     }
```

```
148
149     if (perPtr != NULL)
150     {
151         cout << endl;
152         cout << left;
153         cout << setw(NAME_SIZE) << "NAME";
154         cout << " AGE\n";
155         cout << setw(NAME_SIZE) << string(NAME_SIZE - 1, '-');
156         cout << string(3, '-') << endl;
157         cout << setw(NAME_SIZE) << perPtr->name;
158         cout << " " << perPtr->age;
159         cout << "\n\nIs at the front of the list.\n";
160         cout << right;
161
162     }
163 }
164
165 void List::FindSheep(string name) const
166 {
167     SheepNode* searchPtr;  // CALC    - Searching pointer.
168     bool found;            // CALC    - If search item was found.
169
170     searchPtr = head;
171     found = false;
172
173     if (head != NULL)
174     {
175         cout << endl;
176         cout << left;
177         cout << setw(NAME_SIZE) << "\nNAME";
178         cout << "AGE\n";
179         cout << setw(NAME_SIZE) << string(NAME_SIZE - 1, '-');
180         cout << string(3, '-') << endl;;
181
182         while (!found && searchPtr != NULL)
183         {
184             if (name == searchPtr->name)
185             {
186                 found = true;
187                 cout << setw(NAME_SIZE) << searchPtr->name;
188                 cout << " " << searchPtr->age;
189                 cout << endl;
190                 cout << "Has been found.\n";
191                 cout << right;
192             }
193             else
194             {
195                 searchPtr = searchPtr->next;
196             }
```

```
197         }
198
199         if (!found)
200         {
201             cout << "I\'m sorry, \"" << name << "\" was NOT found!\n";
202         }
203         searchPtr = NULL;
204     }
205     else
206     {
207         cout << "\nCan't search an empty list\n";
208     }
209 }
210
211 void List::ListSize() const
212 {
213     SheepNode* perPtr; // CALC  - Sheep pointer.
214     int         count; // CALC  - List size counter.
215
216     perPtr = head;
217     count = 0;
218
219     while (perPtr != NULL)
220     {
221         count++;
222         perPtr = perPtr->next;
223     }
224
225     if (count == 0)
226     {
227         cout << "Nobody is in the list!\n";
228     }
229     else if (count == 1)
230     {
231         cout << "There is one sheep in the list.\n";
232     }
233     else
234     {
235         cout << "There are " << count << " sheep in the list.\n";
236     }
237 }
238
239 /*****************************************************************************
    **
240 * AddSheep
241 *   This function will add a new sheep to the tail of the linked list.
242 * INPUTS:
243 *   sheep : sheep object.
```

```
244 *
245 * No outputs.
246 *
    ****************************************************************************
    */
247
248 /
    ****************************************************************************
    **
249 * ClearList
250 *   This function will clear the linked list of all sheep.
251 * No inputs.
252 *
253 * No outputs.
254 *
    ****************************************************************************
    */
255
256 /
    ****************************************************************************
    **
257 * DisplayList
258 *   This function will display the entire linked list in a specific format.
259 * No inputs.
260 *
261 * No outputs.
262 *
    ****************************************************************************
    */
263
264 /
    ****************************************************************************
    **
265 * OutputFirst
266 *   This function will display the information of the first sheep in the
    linked
267 *   list.
268 * No inputs.
269 *
270 * No outputs.
271 *
    ****************************************************************************
    */
272
273 /
    ****************************************************************************
    **
274 * FindSheep
275 *   This function will look for the inputed sheep name inside the linked list
```

```
276  *   and output the sheep's information if it was found.
277  *
278  * INPUTS:
279  *  name : Sheep's name to search for.
280  *
281  * No outputs.
282  *                                                                        ↵
       ************************************************************************ ↵
       */
283
284  /                                                                       ↵
       ************************************************************************ ↵
       **
285  * ListSize
286  *   This function will display the size of the linked-list.
287  * No inputs.
288  *
289  * No outputs.
290  *                                                                        ↵
       ************************************************************************ ↵
       */
```