```cpp
/*************************************************************
* AUTHOR    : Ali Eshghi
* STUDENT ID    : 1112261
* LAB #13       : LAB 13 - ARRAYS AND LINKED LIST (OOP)
* CLASS         : CS 1B
* SECTION       : MW - 7:30 pm - 9:50 pm
* DUE DATE      : 12/3/2019
*************************************************************/

#include "MyHeader.h"
#include "ClassHeader.h"


Animal::Animal()/*** CONSTRUCTOR ***/
{
    /*******************
     *  INITIALIZATION *
     *******************/

    name.clear();
    age  = 0;
    listSize = 0;
    head = NULL;
}

Animal::~Animal()    /*** DESTRUCTOR ***/
{

    /*************
     *  VARIABLE *
     *************/

    SheepNode *sheepPtr;

    if(head != NULL)
    {
        //clear the list
        sheepPtr = head;
        while(sheepPtr != NULL)
        {
            head = head -> next;
            delete sheepPtr;

            sheepPtr = head;
        }

        for(int i = 0; i < AR_SIZE; i++)
        {
            nameAr[i] = ' ';
        }
        cout << "The list has been cleared!" << endl << endl;
    }

    else if(head == NULL)
    {
```

```
56          cout << "\nThe list is empty" << endl << endl;
57      }
58
59  }
60
61
62  //method for adding a new sheep and its age to parallel arrays
63  void Animal::AddSheep(string name, int age)
64  {
65      if(listSize < AR_SIZE)
66      {
67          nameAr[listSize] = name;
68          ageAr[listSize]  = age;
69
70          listSize++;
71
72          cout << endl << endl;
73          cout << "The Sheep..."           << endl;
74          cout << "Sheep Name: "   << name << endl;
75          cout << "Sheep Age:  "   << age  << endl;
76          cout << "Has been added" << endl << endl;
77      }
78
79      else
80      {
81          cout << "could not add new animal, list is full..." << endl;
82      }
83
84  }
85
86  //method for adding a new sheep to a linked list
87  void Animal::AddSheepLinkedList(string name, int age)
88  {
89      /*************
90       *  VARIABLE *
91       *************/
92
93      SheepNode *newSheepNode;
94      SheepNode *tail;
95
96      /*******************
97       *  INITIALIZATION *
98       *******************/
99
100     newSheepNode = new SheepNode;
101
102     /*** ADD TO THE TAIL ***/
103
104     //check if there is memory for new node
105     if(newSheepNode != NULL)
106     {
107         newSheepNode -> sheepName = name;
108         newSheepNode -> sheepAge  = age;
109
110
```

```
111         //check if list is empty
112         if(head != NULL)
113         {
114             tail = head;
115 
116             //find the tail
117             while(tail != NULL)
118             {
119                 tail = tail -> next;
120             }
121 
122             tail -> next = newSheepNode;
123         }
124 
125         else
126         {
127             head = newSheepNode;
128         }
129 
130         listSize++;
131     }
132 
133     else
134     {
135         cout << "Could not add to the list - out of memory";
136     }
137 }
138 
139 //method that returns the size of the list of the sheeps
140 int Animal::ListSize() const
141 {
142     if(head != NULL)
143     {
144         return listSize;
145     }
146     else if(head == NULL)
147     {
148         cout << "\nThe list is empty" << endl << endl;
149         return 0;
150     }
151 
152 }
153 
154 //method for showing the first sheep from the list
155 void Animal::DisplayFirstSheep()
156 {
157     if(head != NULL)
158     {
159         cout << left;
160         cout << setw(15) << "NAME" << "AGE" <<endl;
161         cout << setw(15) << "--------------" << "---" << endl;
162         cout << setw(16) << nameAr[0] << ageAr[0];
163         cout << endl << endl << endl;
164 
165         cout << "Is at the front of the list!" << endl << endl;
```

```
166     }
167
168     else if(head == NULL)
169     {
170         cout << "Nobody is in front -the list is empty!" << endl << endl;
171     }
172
173
174 }
175
176 //method for finding the sheep in the list
177 void Animal::FindSheep(string search) const
178 {
179     /*************
180      *  VARIABLE *
181      *************/
182
183     int   i;
184     bool  stat;
185
186     /*******************
187      *  INITIALIZATION *
188      *******************/
189
190     i    = 0;
191     stat = false;
192
193     if(head != NULL)
194     {
195         while(i < AR_SIZE && stat)
196         {
197             if(nameAr[i] == search)
198             {
199                 stat = true;
200             }
201
202             else
203             {
204                 i++;
205             }
206
207             if(stat == true)
208             {
209                 cout << setw(15) << "NAME" << "AGE" <<endl;
210                 cout << setw(15) << "---------------" << "---" << endl;
211                 cout << setw(16) << nameAr[i] << ageAr[i] << endl << endl;
212                 cout << "Has Been Found";
213             }
214         }
215     }
216
217     else if(head == NULL)
218     {
219         cout << "There are no sheep to be found!" << endl << endl;
220     }
```

```
221
222 }
223
224
225 //method  for outputting the objects
226 void Animal::Display() const
227 {
228     /*************
229      *  VARIABLE *
230      *************/
231
232     SheepNode *sheepPtr;
233
234     if(head != NULL)
235     {
236         cout << "<output using the array>" << endl;
237         cout << left;
238         cout << setw(15) << "NAME" << "AGE" <<endl;
239         cout << setw(15) << "---------------" << "---" << endl;
240
241         for(int index = 0; index < listSize; index++)
242         {
243             cout << setw(16) << nameAr[index] << ageAr[index] << endl;
244         }
245         cout << endl << endl;
246
247
248         cout << "<output using the linked list>" << endl;
249         cout << left;
250         cout << setw(15) << "NAME" << "AGE" <<endl;
251         cout << setw(15) << "---------------" << "---" << endl;
252
253         for(sheepPtr = head -> next; sheepPtr != NULL; sheepPtr = sheepPtr ->
    next)
254         {
255             cout << setw(16) << sheepPtr->sheepName << sheepPtr->sheepAge << endl;
256         }
257
258     }
259
260     else if(head == NULL)
261     {
262         cout << "\nThe list is empty" << endl << endl;
263     }
264
265 }
266
267
268
269
```