```
1  /*
2      Nguyen, Da
3      Banh, Alex
4
5      CS A250
6      February 19, 2018
7
8      lab 4
9  */
10
11 #include "AnyList.h"
12
13 //deleteNode
14 void AnyList::deleteNode(int deleteData)
15 {
16     /*
17         PSEUDOCODE:
18
19         IF the list is emtpy
20             print an error message, "Cannot delete from an empty list."
21
22         ELSE
23             Create pointer current to traverse the list, and start by
24             pointing it to the first node.
25             Set a Boolean value to keep track of whether the item is found
26             or not.
27
28             IF the node to be deleted is the first node in the list
29                 Connect the list object to the second node in the list.
30                 Delete the first node.
31                 Set current to NULL.
32                 Decrement the count.
33                 Set the Boolean value to true (you found the node to delete).
34             ELSE (keep searching the list)
35                 Create a pointer trailCurrent to point to the first node
36                 (you already know that the first node is not the one that
37                  needs to be deleted.)
38                 Make pointer current point to the second node.
39
40                 WHILE there are still nodes to check and the node has not been
                     found
41                   IF the node is found
42                       Connect the node to which trailCurrent is pointing
43                       to the node after the node current is pointing to.
44                       (This would work also if current is pointing to the last
                      node. Why?)
45                       Delete the node current is pointing to.
46                       Re-set both current and trailCurrent to NULL.
47                       Decrement the count.
48                       Set the Boolean value to true (you found the node to
                      delete).
49                   ELSE
```

```
50                          Move trailCurrent and current forward.
51
52              IF the node is not found
53                  print out the error message, "Item to be deleted is not in the
                      list."
54      */
55
56      if (ptrToFirst == nullptr)
57          cerr << "Cannot delete from an empty list." << endl;
58      else
59      {
60          Node *current = ptrToFirst;
61          bool itemFound = false;
62
63          if (ptrToFirst->getData() == deleteData)
64          {
65              ptrToFirst = ptrToFirst->getPtrToNext();
66              delete current;
67              current = NULL;
68              --count;
69              itemFound = true;
70          }
71          else
72          {
73              Node *trailCurrent = ptrToFirst;
74              current = current->getPtrToNext();
75
76              while (current != nullptr && !itemFound)
77              {
78                  if (current->getData() == deleteData)
79                  {
80                      trailCurrent->setPtrToNext(current->getPtrToNext());
81                      delete current;
82                      current = NULL;
83                      trailCurrent = NULL;
84                      --count;
85                      itemFound = true;
86                  }
87                  else
88                  {
89                      trailCurrent = current;
90                      current = current->getPtrToNext();
91                  }
92              }
93          }
94
95          if (!itemFound)
96              cerr << "Item to be deleted is not in the list." << endl;
97      }
98  }
```