```cpp
/*
    Salcedo, Salvador

    CS A250
    March 9, 2019

    Lab 6
*/

#include "DoublyList.h"

// createAList
void DoublyList::createAList()
{
    /*
    NOTE:

    * You will need to declare one pointer and
    you may re-use this pointer throughout the function, but
    you are NOT allowed to create additional pointers.

    * DO NOT REMOVE EXISTING COMMENTS.

    * Pay CLOSE attention to instructions.
    */

    /*--------------------------------------------------------------------
    SECTION 1
    --------------------------------------------------------------------*/

    // Create a dynamic node that stores the value 2 and make
    // this node be the first node of the calling object.
    // List becomes: 2
    // Use the overloaded constructor.
    // Do NOT create a pointer.
    first = new Node(2, nullptr, nullptr);
    last = first;

    // Update count;
    count++;

    cout << "SECTION 1 - TEST ALL" << endl;
    testAll();

    /*--------------------------------------------------------------------
    SECTION 2
    --------------------------------------------------------------------*/

    // Create another node that stores the value 3 and
```

```
50      // insert this node to the left of the node that is
51      // storing value 2.
52      // List becomes: 3 2
53      // Do NOT create a pointer.
54      first->setPrev(new Node(3, nullptr, first));
55      first = first->getPrev();
56
57      // Update count;
58      count++;
59
60      cout << "\nSECTION 2 - TEST ALL" << endl;
61      testAll();
62
63      /*-------------------------------------------------------------
64      SECTION 3
65      -------------------------------------------------------------*/
66
67      // Create another node that stores the value 4 and
68      // insert this node to the right of the node that is
69      // storing value 3.
70      // List becomes: 3 4 2
71      // NO MORE than 3 statements.
72      first->setNext(new Node(4, first, last));
73      last->setPrev(first->getNext());
74
75      // Update count;
76      count++;
77
78      cout << "\nSECTION 3 - TEST ALL" << endl;
79      testAll();
80
81      /*-------------------------------------------------------------
82      SECTION 4
83      -------------------------------------------------------------*/
84
85      // Delete the first node.
86      // List becomes: 4 2
87      Node * temp = first;
88      first = first->getNext();
89      first->setPrev(nullptr);
90      delete temp;
91      temp = nullptr;
92
93      // Update count.
94      count--;
95
96      cout << "\nSECTION 4 - TEST ALL" << endl;
97      testAll();
98
```

```
 99      /*-------------------------------------------------------------
100      SECTION 5
101      -------------------------------------------------------------*/
102
103      // Insert three nodes at the end of the list storing
104      // 5 6 7 in this order.
105      // List becomes: 4 2 5 6 7
106      // Do NOT use the pointer you created.
107      last->setNext(new Node(5, last, nullptr));
108      last = last->getNext();
109      last->setNext(new Node(6, last, nullptr));
110      last = last->getNext();
111      last->setNext(new Node(7, last, nullptr));
112      last = last->getNext();
113
114      // Update count.
115      // One statement only.
116      count += 3;
117
118      cout << "\nSECTION 5 - TEST ALL" << endl;
119      testAll();
120
121      /*-------------------------------------------------------------
122      SECTION 6
123      -------------------------------------------------------------*/
124
125      // Move last node to second position.
126      // Here steps are very important. Carefully think
127      // how you can move nodes around without losing any
128      // nodes and keeping all pointers pointing to the
129      // correct nodes.
130      // Note:
131      //      You may NOT create an additional node.
132      //      NO loops are necessary.
133      // List becomes: 4 7 2 5 6
134      temp = last;
135      last = last->getPrev();
136      last->setNext(nullptr);
137      first->getNext()->setPrev(temp);
138      temp->setNext(first->getNext());
139      first->setNext(temp);
140      temp->setPrev(first);
141
142      cout << "\nSECTION 6 - TEST ALL" << endl;
143      testAll();
144
145      /*-------------------------------------------------------------
146      SECTION 7
147      -------------------------------------------------------------*/
```

```
148
149     // Move the first node in between the node before last and
150     // last node (the second node will become the first node
151     // in the list, and the first node will become the before-last
152     // node in the list).
153     //      You may NOT create an additional node.
154     //      No loops are necessary.
155     // List becomes: 7 2 5 4 6
156     temp = first;
157     first = first->getNext();
158     first->setPrev(nullptr);
159     last->getPrev()->setNext(temp);
160     temp->setPrev(last->getPrev());
161     temp->setNext(last);
162     last->setPrev(temp);
163
164     cout << "\nSECTION 7 - TEST ALL" << endl;
165     testAll();
166
167     /*-------------------------------------------------------------
168     SECTION 8
169     -------------------------------------------------------------*/
170
171     // WITHOUT moving any nodes, swap around the values to
172     // create an ordered list.
173     // Note that there is no need to move the value 5.
174     // You may declare an int, BUT do NOT use any literals.
175     // List becomes: 2 4 5 6 7
176
177     int store = first->getData();
178     first->setData(first->getNext()->getData());
179     first->getNext()->setData(store);
180     first->getNext()->setData(last->getPrev()->getData());
181     last->getPrev()->setData(last->getData());
182     last->setData(store);
183
184     cout << "\nSECTION 8 - TEST ALL" << endl;
185     testAll();
186
187     /*-------------------------------------------------------------
188     SECTION 9
189     -------------------------------------------------------------*/
190
191     // Add two nodes storing 1 and 3 to complete the ordered list.
192     // List becomes: 1 2 3 4 5 6 7
193     temp = first->getNext();
194     first->setPrev(new Node(1, nullptr, first));
195     first = first->getPrev();
196     temp->setPrev(new Node(3, first->getNext(), temp));
```

```
197     first->getNext()->setNext(temp->getPrev());
198     // Add 2 to count.
199     count += 2;
200
201     cout << "\nSECTION 9 - TEST ALL" << endl;
202     testAll();
203
204     /*-----------------------------------------------------------------
205     SECTION 10
206     -----------------------------------------------------------------*/
207
208     // Go back to check the following:
209     //     Are there any sections that have more than one blank line?
210     //     If so, do not leave delete unnecessary white space.
211     //     Leave only one blank line.
212     //
213     //     Are your statement too long that is necessary to scroll horizontally?
214     //     If so, break your statements in readable portions.
215     //
216     //     Instructions said to create and use ONLY ONE pointer.
217     //     Go back and check that you did not create more than one pointer.
218     //
219     //     Re-visit your code for efficiency.
220
221 }
```