```cpp
/*
    Nguyen, Da
    Banh, Alex
    Ton, An

    CS A250
    March 3, 2018

    Lab 5
*/

#include "DoublyList.h"

// Definition function print
void DoublyList::print() const
{
    Node *current = first;  //create a current pointer pointing to the first node

    while (current != nullptr)
    {
        cout << current->getData() << "  ";
        current = current->getNext();
    }
}

// Definition function reversePrint
void DoublyList::reversePrint() const
{
    Node *current = last;

    while (current != nullptr)
    {
        cout << current->getData() << "  ";
        current = current->getPrev();
    }
}

// Definition function front
int DoublyList::front() const
{
    return first->getData();
}

// Definition function back
int DoublyList::back() const
{
    return last->getData();
}

// Definition function copyToList
void DoublyList::copyToList(DoublyList& otherList) const
{
```

```
53      Node *temp = last;
54
55      while (temp != nullptr)
56      {
57          otherList.insertFront(temp->getData());
58          temp = temp->getPrev();
59      }
60  }
61
62  // Definition function insertInOrder
63  void DoublyList::insertInOrder(int insertElement)
64  {
65      //if the list is empty or insertElement is "<" the data of first node.
66      if (first == nullptr || insertElement < first->getData())
67      {
68          //Using the insertFront() to insert the new node in front of first node.
69          insertFront(insertElement);
70      }
71      else
72      {
73          //create newNode pointer points to new node
74          Node *newNode = new Node(insertElement, nullptr, nullptr);
75
76          //if the data in the last node is "<" insertElement.
77          if (last->getData() < insertElement)
78          {
79              last->setNext(newNode);     //connect from last to newNode.
80              newNode->setPrev(last);     //connect from newNode to last.
81              last = newNode;     //move last pointer pointing to newNode.
82              count++;
83          }
84          else
85          {
86              //create the current pointer travelling the list.
87              Node *current = first->getNext();
88              bool found = false;     //keep tracking the data on the list.
89
90              while (current != nullptr && !found) //search the list.
91              {
92                  //if the data at current is ">=" insertElement.
93                  if (current->getData() >= insertElement)
94                  {
95                      //newNode points to current.
96                      newNode->setNext(current);
97                      //newNode points to prev-node from the node at current.
98                      newNode->setPrev(current->getPrev());
99                      //the previous node points to newNode.
100                     current->getPrev()->setNext(newNode);
101                     //the node at current points to newNode.
102                     current->setPrev(newNode);
103
104                     count++;
```

```
105                     found = true;
106                 }
107                 else
108                     current = current->getNext();   //current moves to next node.
109             }
110         }
111     }
112 }
```