ENGR:2730 Computers in Engineering
Christensen + Bell Spring 2022, Homework #8

Deadline: Friday, May 6, 2022 by 11:59pm
Early turn in bonus deadline: Wednesday, May 4, 2022 by 11:59pm

Implement a doubly linked list class of integers. You will write two new classes, **Node** and **MyList**. These classes must be implemented in their own files, i.e., Node.cpp, Node.h, MyList.cpp and MyList.h.

Specifications:

1. **[25pts]** Write a main program that tests each method of the **Node** and **MyList** classes. You should implement a class method, test it, and then check your code into SVN. Repeat until finished with homework. ☺

2. **[25pts]** Implement and document a **Node** class.
    a. Your Node class must have three private member variables. These variables should store an integer, a next Node pointer, and a previous Node pointer.
    b. Your Node class must have an explicit constructor to initialize all the private member variables. The constructor must provide default values for each member variable.
    c. Implement getters and setters for all private member variables.
    d. Implement a **print()** method that prints the integer stored in the node.

3. **[50pts]** Implement a **doubly-linked** list of sorted (ascending) integers class called **MyList.**
    a. This class must have exactly one private member variable that points to the current node in the list. For example, you may want to name this variable **current**, **currentPtr**, **m_current**, or **m_currentPtr**. The type of this variable should be **Node ***. This means that **MyList** should not have a **startPtr** (or equivalent).
    b. Your **MyList** class must define a default constructor (i.e., no inputs) that initializes the list to an empty list. In other words, the current pointer should be initialized to the **nullptr**.
    c. Your **MyList** must define a **printAscending()** method. This method should print the integers stored in the list in ascending order. Print that the list is empty if the list is empty. **Pseudocode is provided in the appendix for reference.**
    d. Your **MyList** must define a **printDescending()** method. This method should print the integers stored in the list in descending order. Print that the list is empty if the list is empty.
    e. Your **MyList** must define a **insert(int value)** method. This method must insert the number into the list in ascending order. This method should use the **new** command to create a new node and check to make sure the new command successfully created a new Node. **Pseudocode is provided in the appendix for reference.**
    f. Your **MyList** must define a **remove(int value)** method. This method removes a single node from list that matches the input value if it exists. When removing the node from the list, remember to **delete** it (i.e., to ensure no memory leaks). If list is empty, print that the list was empty and there was nothing to delete. If the value was not found, print a message saying that number was not found in the list. Otherwise, print that the node was deleted.
    g. Your **MyList** must define a **clear()** method. This method should remove all elements from the list. Print the value of each node that is deleted.
    h. Your **MyList** class should define a destructor that deletes all the nodes from the list. The destructor should print the value of each node that is deleted. Hint: use your clear() method.

Submission Instructions:

You must make a CLion project called "hw8" under your SVN homework directory. Check your homework into SVN.

Hint: you can see the current version of your submission by opening this link in a web browser:
https://class-svn.engineering.uiowa.edu/cie/projects/spring2022/

Appendix: Pseudocode for MyList Member Functions

```
printAscending()
{
   If the list is empty
         Print the list is empty and return
   endIf

   Move the current pointer to the first node of the list

   Print the int value stored in each node from the first node to the last node in the list
}

printDescending()
{
   // 👁️👄👁️
}

insert(int value)
{
   create a node for the passed in parameter, value

   if the list is empty
      Set currentPtr to point to the new node
   else
      Find location to insert the new node (may be to the left or the right of the currentPtr)
      Insert the new node into the list
   endIf
}

remove(int value)
{
   // 👉🥺👈
}
```