Assignment #6

Purpose

The purpose of this assignment is to continuing practicing creating dynamic structures. You will be creating a doubly-linked list.

Instructions

In this lab you will create a class to implement a doubly linked list. You will implement the following concepts:

- Create the List
- Clear the List
- Delete the List
- Add Node to the Front of the List
- Add Node to the Back of the List
- Get the Contents of the First Node
- Get the Contents of the Last Node
- Remove the First Node
- Remove the Last Node
- Remove a Node from the list that has a given value
- Remove all Nodes from the list that have a given value
- Find if a value exists in the list
- Get the Number of Nodes in the list
- Return a string of all items in the list, either forwards or backwards

You will implement two classes for your doubly-linked list

Class 1 - DLList

This class for your doubly-linked list.

Private Data Members:	The size of your list Pointer to the head node Pointer to the tail node
Constructor:	Initializes head and tail to **`NULL`**, size to **`0`**.
Destructor:	Calls the **`Clear()`** Function
Member Function 1:	Named **`GetSize`**. Returns the size of the list. **`const`** function.

Member Function 2:	Named **`PushFront`**. Has one parameter, an integer to add to the list. Creates a **`DLNode`** with the contents of the parameter and adds that node to the front of the list.
Member Function 3:	Named **`PushBack`**. Has one parameter, an integer to add to the list. Creates a **`DLNode`** with the contents of the parameter and adds that node to the end of the list.
Member Function 4:	Named **`GetFront`**. Returns the integer value of the node at the beginning of the list. **`const`** function. Output **`"List Empty"`** to standard error output if the list is empty and return 0.
Member Function 5:	Named **`GetBack`**. Returns the integer value of the node at the end of the list. **`const`** function. Output **`"List Empty"`** to standard error output if the list is empty and return 0.
Member Function 6:	Named **`PopFront`**. Removes the first node in the list. Output **`"List Empty"`** to standard error output if the list was already empty.
Member Function 7:	Named **`PopBack`**. Removes the last node in the list. Output **`"List Empty"`** to standard error output if the list was already empty.
Member Function 8:	Named **`RemoveFirst`**. Has one parameter, an integer to find. If the value is found, remove that node from the list. Will only remove the first node that matches the value. If the value is not found, output **`"Not Found"`** to standard error output.
Member Function 9:	Named **`RemoveAll`**. Has one parameter, an integer to find. If the value is found, remove that node from the list. Will remove all nodes that match the value. If the value is not found, output **`"Not Found"`** to standard error output.
Member Function 10:	Named **`Exists`**. Has one parameter, an integer to find. If the value is found, return **`true`**, else return **`false`**.
Member Function 11:	Named **`Clear`**. Clears all nodes from the list, resets the size to 0 and head and tail to **`NULL`**.

Member Function 12:	Named **`ToStringForwards`**. Outputs the contents of the list as a comma separated list ("1, 2, 3, etc") starting at the first node and ending at the last node. If the list is empty return the empty string and output **`"List Empty"`** to standard error output.
Member Function 13:	Named **`ToStringBackwards`**. Outputs the contents of the list as a comma separated list ("1, 2, 3, etc") starting at the last node and ending at the first node. If the list is empty return the empty string and output **`"List Empty"`** to standard error output.

Class 2 - DLNode

Models information about a single node in the **`DLList`**

Private Data Members:	An integer for the contents of the node A pointer to the previous node A pointer to the next node
Constructor #1:	No parameters Sets the integer to 0 Sets the previous node to **`NULL`** Sets the next pointer to **`NULL`**
Destructor:	Is empty.
Mutator 1:	Named **`SetContents`**. Has one integer parameter. Sets the internal integer to the given parameter's value.
Mutator 2:	Named **`SetNext`**. Has one parameter which is a pointer to a DLNode. Sets the internal next pointer to the given parameter's value.
Mutator 3:	Named **`SetPrevious`**. Has one parameter which is a pointer to a DLNode. Sets the internal previous pointer to the given parameter's value.
Accessor 1:	Named **`GetContents`**. Returns the value of the internal integer. **`const`** function.

Accessor 2:	Named **`GetNext`**. Returns the pointer to the next node. **`const`** function.
Accessor 3:	Named **`GetPrevious`**. Returns the pointer to the previous node. **`const`** function.

Objectives

- Implement and perform a complete set of operations on a doubly-linked list
- Improve skills of memory management and pointer syntax

Requirements

Your code must follow the styling and documenting guidelines presented in class. Please note that I do not give points for style and documentation. You can only lose points. Please make sure your source code is documented correctly and is neatly and consistently formatted using guidelines provided in class.

Your program must provide the features described above:

[50 pts] - Fully complete **`DLNode`**

[100 pts] - Fully complete **`DLList`**

IF YOUR PROGRAM DOES NOT COMPILE YOU WILL RECEIVE A ZERO!!!

Deliverables (via Blackboard)

Your files need to be uploaded/attached to the Assignment Submission on Blackboard. You should upload the following four files:

- **`dl_node.h`**
- **`dl_node.cpp`**
- **`dl_list.h`**
- **`dl_list.cpp`**