TCES 203
Programming Practicum
Assignment 7 – Exceptions and Class templates
15 Points

This assignment tests your understanding of concepts covered in the course dealing with exceptions and class templates. Revisit four classes and the main function to add exception handling and make classes support any data type with class templates.

You will create a doubly linked list that contains a head and tail node as member variables. You will create two derived classes called Stack and Queue that use the doubly linked list as a base class. Test each class thoroughly before moving on to the next class. Don't forget to check for memory leaks.

A **Node** class that allows user to store any one single piece of data and stores a reference to the previous and the next node. Provide a constructor, overloaded constructor and setters/getters for the member variables. Make your member variables private.

A **List** class that represents a doubly linked list with a reference to the front and the back node of the list. Provide a default and overloaded constructor that takes any data type as the parameter. Provide addFirst, addLast, removeFirst and removeLast functions to allow for addition and removal of data. Provide a size function that returns the size of the list as well as an isEmpty function that returns true or false. Overload << operator to print the contents of the list. Provide a destructor. Throw an exception if the list is empty and any of the remove functions are called.

A **Stack** class that inherits from the List class. Provide push, pop functions that follow the stack policy. Overload << operator to print the contents of the stack. The function pop should output that the stack is empty if you call the function on an empty stack. Throw an exception if the stack is empty and pop function is called.

A **Queue** class that inherits from the List class. Provide enqueue, dequeue functions that follow the queue policy. Overload << operator to print the contents of the queue. The function dequeue should output that the queue is empty if you call the function on an empty queue. Throw an exception if the queue is empty and dequeue function is called.

Overloading << operator with inheritance will pose a problem since it is a friend function. A way to get around this problem is to provide a virtual member function that will print the corresponding class members (to ostream) in the base class and the corresponding derived classes. This member function can then be called from the operator function.

Submission and Grading:

Node.h, Node.cpp, List.h, List.cpp, Stack.h, Stack.cpp, Queue.h, Queue.cpp and main.cpp files must be on Subversion as we did in the lab under the Project name, ExceptionsAndTemplates. You still have only till midnight to do this. Please do not commit past midnight.

There will be points taken off for not following the conventions listed in this document regarding submissions, outputs and naming conventions.

You are required to properly indent your code and will lose points if you make significant indentation mistakes. See the textbook for an explanation and examples of proper indentation.

Give meaningful names to functions and variables in your code. Localize variables whenever possible -- that is, declare them in the smallest scope in which they are needed.

Include a comment at the beginning of your program with basic information and a description of the program **and include a comment at the start of each function**. Your comments should be written in your own words and not taken directly from this document. Write comments within functions to explain the flow or any obscure code. Provide comments for the functions as well as for the class definition. Make sure that every file has a header comment including the .h and the main.cpp files.

You should include a comment at the beginning of your program with some basic information and a description of the program, as in:

```
// Menaka Abraham
// 3/30/15
// 203
// Assignment #1
//
// This program will...
```