CS 133 A

Project 2: "Circular Doubly-Linked List Queue"

Name:_____

In class, you were given an implementation of a **linked list.** Your task is to now modify that code to turn it into a doubly-linked circular list, and to test your new creation. A **doubly-linked list** is made up of nodes that contain a reference to the ***previous*** node as well as the ***next*** node--allowing for traversal in ***either*** direction.

A **circular doubly-linked list** has the ***next*** reference of the last node refer to the first node (instead of ***null***), and has the ***previous*** reference of the first node refer to the last node (instead of ***null***)--thus making a circle (*you will need to make your own design decisions about what to do with a list with only one node*).

Your first step should be to make sure that the starting code (the Simple Doubly-linked list Class) works, and that all declared methods are implemented as described on the given web page. You should also make this be a Template class (**CircularDoublyLinkedList<T>**), instead of having just a typedef for datatype. This new class should contain all of the same public methods as in the given class, but should be re-written to take advantage of the capabilities of the new construction. Also, since you will now be using this as a base class, you will need to determine which methods should be ***virtual.***

Many methods will require only minor modification. Others will require more thinking. You may wish to draw diagrams to help you through the planning process.
When this class is fully tested, you will need to write the definition for a new **queue** class which ***protectedly*** inherits from your Circular Doubly-Linked List class. It should contain only:

- constructors to match up with all of the constructors in the Circular Doubly-Linked List class.
- a destructor
- **getSize(), begin(), end(), empty(), release()**
- **void push(T& element)**
- **T& pop()**

The structure of the underlying list must stay the same (**a *circular* doubly-linked list**). An **iterator** should still work exactly the same way. If methods act the same as the parent-class method, they should merely call the parent-class method (for constructors, use the initialization list).

You will also need to write a program which is a test driver for the classes--that is a main that tests the methods of both classes. It should contain appropriate ***try/catch*** blocks and test all of the methods of both classes, either directly or indirectly. Protected or private inheritance makes it hard to test ***virtual*** functions, so you do not need to worry about this here. You may use any class you choose that makes sense as the data implementation (***string*** *is an easy choice*).

The program should be fully planned *in advance*. It should be properly documented, and work correctly.

Extra Credit: (*10 points*) If you wish, you may rewrite the queue class (in a separate project) so that it inherits **publicly** from ***CircularDoublyLinkedList.*** You should then ***override*** any methods you do not wish to use so that they merely throw an exception stating that the method is not supported (write your own child class of exception called "*methodNotSupported*"), or you may override with "*=delete*". Test using a ***CircularDoublyLinkedList pointer*** to point to objects of ***BOTH*** types, to see if any ***virtual*** functions behave properly. It is up to you how it goes about doing this.

Possible: 100 points

Deliverables:

Physical:

The Project should be turned in inside a clear plastic file folder. This folder should have a simple flap to hold paper in place--NO buttons, strings, Velcro, etc. Pages should be in order, not stapled.

- Assignment Sheet (printed pdf from the web), with your name written on it, as a cover sheet.
- Printed Source Code with Comments *(including heading blocks -- a file header for each file plus a function header for each function. Describe parameters, any input or output, etc., no line wrapping). Print in portrait mode, 10 - 12 point font.*

Electronic:

- All .h, .cpp, .exe(Release Version) zipped together. Do not use rar or any archive format other than zip. Rename the file: "<YourName>_p2.zip".
- Sample Output (as .rtf -- run the program, copy the window using <*Alt|PrtScn*>, paste into Paint, invert colors *(<Ctrl|Shift|I*>), copy, open Wordpad, save.)
- A simple test plan including explanations of any discrepancies and reasons for each test. Show actual input and ALL values output as well as ALL expected output. Test each possible action. Save as .xls, xlsx, .doc or .docx file
- Submit this single zip file by going to canvas, select this class, select the Assignment tab on the left, select Assignment 2, select the submission tab at the top right, find the file, and Submit.

Due: Friday, May 20, 2016 9:30 a.m. *(beginning of class)*