Due: 7/19 (11:59PM)

Let us code the following three source files in C++:

- **StringList.h**
  - In this file, you declare a class named *StringList*.
  - *StringList* is a modified version of *NumberList* class (Chapter 17), that is designed to store C++ strings in a linked list.
  - Therefore, each node must store a `string`, not a number.
  - The node structure must be declared within class, and a private attribute *head* points to the starting node.
  - The class provides a default constructor that creates an empty linked list, and a destructor that releases all the nodes.
  - The class also provides public member functions for inserting and deleting a node (see below for details).
  - It also provides a public member function for displaying the list.
- **StringList.cpp**
  - In this file, you provide definitions for the default constructor and the destructor for *StringList*.
  - Make sure the destructor visits every node in the list and deletes every one of them from the heap.
  - Define *insertFront* function to insert a new node into the front of the list. Therefore, the new node will be the new first node in the list. This function takes a `string` as a parameter.
  - Define *insertBack* function to insert a new node into the back of the list. Therefore, the new node will be the new last node in the list. This function takes a `string` as a parameter.
  - Define *deleteFront* function to delete the first node from the list. This function takes no parameter.
  - Define *deleteBack* function to delete the last node from the list. This function takes no parameter.
  - Define *display* function that displays the current contents (strings) of the list (display strings in a single line, separated by a space). This function takes no parameter.
- **hw5.cpp**
  - In this file, you define *main* function that tests *StringList* class.
  - You must first create a *StringList* object.

- Then start inserting new nodes, one at a time. Alternate between inserting into the front and into the back. Make sure to add at least 6 nodes.
- Then start deleting nodes, one at a time. Alternate between deleting from the front and from the back.
- Make sure all the nodes are deleted before terminating the program.
- After each insertion or deletion, call *display* member function to display the updated list.
- Make sure all the member functions are tested and shown to work properly, without missing any of them.
- Note that the *display* function must be called pretty much every time a node is inserted or deleted. Otherwise the respective functionality will not be properly demonstrated and *could lead to loss of points*.
- Also note that each member function should properly work regardless of the current list configuration (empty list, one-node list, or multiple-node list).

What to submit:

- All source files (.cpp, .h) needed for compilation (do not submit the entire Project)

How to submit:

- Use Canvas Assignment Submission menu to submit the assignment electronically at Canvas.
- Make sure to zip all your files into `hw4.zip`, then submit your `hw4.zip` as a single file.

Policy

- Make sure all your C++ programs properly compile and run on Eclipse C++.
- Projects will be graded 20% on style/standards and 80% on proper execution. Make sure to follow the *Coding Standards* posted on the course webpage. In particular, use proper indentation on each line of your code.
- At the beginning of each file (.cpp, .h), provide comments specifying the author, date, and a brief description of the file.
- Each source file (.cpp, .h) must contain enough comments here and there to make it easy to follow your code. Insufficient comments could lead to loss of points.
- Non-compilable program will get almost no credit (e.g., executable code not produced due to compile errors).
- Non-working program will get almost no credit (e.g., the executable is terminated immaturely due to run-time errors).
- Copying other's code is strictly prohibited. If identical (or nearly identical) submissions are found among students, every student involved will get automatic zero for the assignment. The same goes for copying existing source code from the internet.