Project 1

For this project we are going to use a linked list to implement a new data structure that we have not covered in class but are similar to ones that we have covered in detail. Before you begin, you must write yourself a `LinkedList` class and a `Node` class (please name your classes **exactly** as I did here). Please follow the below specifications for the two classes.

`Node.cpp`

- This must be a generic class.
- Should contain a generic member variable that holds the nodes value.
- Should contain a `next` and `prev` `Node*` as denoted here.
- All member variables should be private.
- Use `public` and `private` member functions as you deem fit.

`LinkedList.cpp`

- This class should be the only class creating and manipulating `Node` objects.
- This class must be able to handle (1) singly, (2) doubly, (3) singly-circularly, and (4) doubly-circularly linked list. When the list is created, it must be given input to which type of list it will be.
- This class must also have generic functions that are able to properly handle the generic node values.
- Your class must at least have the following `public` prototypes implemented.
  - `bool perform_operation(int op_code, T value);`
- Your class must at least have the following `private` prototypes implemented.
  - `void append(Node* n);            (Opcode 1)`
  - `void remove_tail();              (Opcode 2)`
  - `void push(Node* n);              (Opcode 3)`
  - `Node* pop();                     (Opcode 4)`
  - `void remove_i(int i);            (Opcode 5)`
  - `Node* get_i(int i);              (Opcode 6)`
  - `void add_i(Node* n, int i);      (Opcode 7)`
  - `void print();                    (Opcode 8)`
  - `void delete();                   (Opcode 9)`
  - `void reverse();                  (Opcode 10)`
  - `void rotate(int rotations);      (Opcode 11)`
  - `void random_shuffle();           (Opcode 12)`
  - `int size();                      (Opcode 13)`
- I am not specifying all the little details in the error checking, but remember we would like to write code that does not behave unexpectedly without informing the user of error.
- The perform_opertation function should be the only function the uses the private member functions of the class. It will also be responsible for creating the nodes in memory.

Using the `LinkedList` class you just created, you will need to implement a priority stack through the creation of the `PriorityStack` class. A priority stack is similar to a stack but as in the priority queue we will pop the higher priority items first. You will need to write a main function that will test the functionality of all of your classes. I will also supply a main function that you can use to test your code.

Submission Instructions

1. For this assignment you must submit one .zip file that is named <username>_project_1.zip. All other files must be named based on the class.
2. All source files must compile and run on the cs.ramapo.edu server.
3. E-mail submissions are not accepted.

Assignment Notes

1. Work early and often. Understand the problem first, then develop an algorithm, finally implement. Compile early and often.
2. If the name convention is not followed, the assignment will be **penalized 50%.**
3. I will use the timestamp in Moodle to determine submission time. I do not have access to a history of your submissions. This means that if you resubmit after the deadline it will be marked as late by Moodle. Please do not resubmit after the deadline.
4. As always please **come see me right away** if you are having troubles with any parts of the assignment. The goal of these assignments are to improve your C++ and problem solving skills.

I try my best to avoid errors, however, if you see any please let me know so I can send corrections to the class.