CS 180: Data Structures, Spring 2011
Homework 3

Due *via email* by 11:59pm on Saturday, Feb. 26

1. (5 points) Assume that we have an initially empty stack of integers. Fill in the following table, showing the output returned and the internal state of the stack after each call. (We've filled in the first 2 lines for you.)

Operation	Output	Contents of S
S.push(7)	-	(7)
S.push(10)	-	(7, 10)
S.push(4)		
S.top()		
S.pop()		
S.size()		
S.pop()		
S.empty()		
S.push(11)		
S.pop()		
S.top()		
S.push(3)		

2. (5 points) Assume that we have an initially empty queue of integers. Fill in the following table, showing the output returned and the internal state of the queue after each call. (We've filled in the first 2 lines for you.)

Operation	Output	Contents of Q
Q.push(7)	-	(7)
Q.push(10)	-	(7, 10)
Q.push(4)		
Q.top()		
Q.pop()		
Q.size()		
Q.pop()		
Q.empty()		
Q.push(11)		
Q.pop()		
Q.top()		
Q.push(3)		

3. (5 points) Consider the doubly linked list class given in Section 3.3 of the textbook, and note that the class given in the textbook is missing the housekeeping functions. Write the code for the copy constructor for this class, ensuring that it makes a deep copy of the input object.

4. (10 points) Imagine you are asked to write a program as follows, with only access to the queue and stack data structures from the standard template library. (Note: This means you can use the constructor, the size function, etc., as needed.)

   You want to write a function called Halves which takes as input a reference to an integer queue. This function should break the collection of elements inside the queue into two equal-sized pieces. If there are an odd number of elements in the queue, the first half should contain the extra element.

   The queue should be changed so that the first and second half of the queue are swapped, and the second half is reversed. For example, given the following queue:

```
front                          rear
10  -2  0  3  5  7  2  -8  3  4  14
```

   you want to change the queue into the following:

```
 front                                rear
14  4  3  -8  2  10  -2  0  3  5  7
---------------  ------------------
   reversed          first
   second            half
   half
```

   Just below, we have begun the code for this function, including a few local variables that you may use (but are not required to use - they are just in case you need them). The *ONLY* other variables you can create and use are new stacks and queues.

   After writing this function, please write a main which will also test execution of the function on the example given above.

```
void Halves(queue<int>& Q) {
   int temp1, temp2, temp3;
   // your code goes here
```

5. Extra Credit:

   (3 points) This problem explores variants in argument passing. What (if anything) is different about the behavior of the following three functions f(), g() and h()?

```
void f(int x) {
  x = x + 1;
  cout << x;
}

void g(int& x) {
  x = x + 1;
  cout << x;
}

void h(const int& x) {
  x = x + 1;
  cout << x;
}
```