Homework 3

100 Points

STACKS and QUEUES ADT (template)

Code Review: Read and reuse stack & queue class examples (programs)

Project: Write a stack and queue test driver. A test driver is a program created to test functions that are to be placed in a library. Its primary purpose is to completely test functions therefore it has no application use. You will use two stacks and two queues in the program, as described below.

a. inS – input stack: used to store all user input
b. inQ – input queue: used to store all user input
c. outS – output stack: used to store data deleted from inQ
d. outQ – output queue: used to store data deleted from inS

Use a menu-driven interface that prompts the user to select one of the following options: I – insert, D – delete, C – display the number of elements in the two stacks and two queues, T – display the elements at the top of the two stacks, F – display the elements at the front of the two queues, R – display the elements at the end of the two queues, Q – to quit the program. If an insert is requested, the system should prompt the user for the number (a double) to be inserted. The number is then inserted into the input stack and input queue. If a delete is requested, the data are deleted from both structures: data popped from the input stack are enqueued in the output queue, and the data dequeued from the input queue are pushed into the output stack. Processing continues until the user enters 'Q'. At this point print the contents of the stacks and queues while deleting all data.

Test your program with the following operations:

```
D
I 10.5
I 20.1
D
I 30.2
D
I 40.1
I 50.5
C
T
F
R
D
I 60.5
D
I 70.9
D
I 80.2
D
D
D
Q
```

Save the output as a comment at the end of the source file containing main().

The stack library is incomplete: you have to implement the **`getTop`** (or **`stackTop`** or **`peek`**) function that passes back the data on top of the stack, without changing the stack, and the **`getCount`** (or **`stackCount`**) function that returns the number of elements in the stack.

Run the program once and save the output at the end of the source file as a comment. Compress the source and header files, input and output files (if any), and your report, and upload the compressed file: 22C_LastName_FirstName_H3.zip

Next Page

Grading:

Implement the **`getTop`** function	- 5
Implement the **`getCount`** function	- 5
Read input	- 5
Test stack functions	- 30
Test queue functions	- 30
Program structure (write sub-functions)	- 15
The main() function (no memory leak)	- 5
Self Assessment Report	- 5

Self Assessment Report: Write a short report, (see `22C_H_3Report.doc` form) briefly explaining your code and containing an assesment of your implementation based on the above grading criteria.

Note: See class examples in Stack and Queue Demo folders on Canvas. In your program you will use the stack and queue template given in class. The stack template library is incomplete: you have to implement the **`getTop`** function that passes back the data at the top of the stack, without changing the stack, and the **`getCount`** function that returns the number of elements in the stack. *Class templates are used to create generic classes and abstract types. They enable you to create one general version of a class without replicating code to handle multiple data types.*