COSC2430 HW6: Class-Stack-Queue Problem

1. Introduction

My friend SSS is the owner of a bookstore. She buys and sells books. After buying, she can store books either in the shelf maintaining the queue, or piling books one upon one using the stack. But the problem is, during selling the book, she needs to move books from the top of the stack, or from the front of the queue to get the specific book. Find out the cost, number of moves, that she needs for the stack, and for the queue to get the specific book.

Note: It is required to implement the Stack and Queue by yourself. See the Implementation section to be clear how to implement it.

2. Input and Output

The input and output file is a regular text file, where each line is terminated with a '\n' character.

Input file format:

- Each line is a combination of a command and a book name.
- Command and book name is separated by a single space.
- The command can be only either “buy” or “sale”.
- If there is an empty line, you should ignore it.

Output file format:

- Your program will generate output only for “sale” command.
- For a successful sale command, the output will be like,
  Book_name finding cost at stack: *value*, at queue: *value*
- For an unsuccessful sale command, the output will be like,
  Book_name not found

You know, Stack follows LIFO, last in first out, order, and Queue follows FIFO, first in first out, order. You will add book into both stack and queue after buying. To

get a book, in case “sale” command, you have to move books one by one from the stack, pop(), and the queue, dequeue(). After getting your sold book, you have to again insert other books into the stack, push(), in a ***reverse order***, and into the queue, enqueue(), in a ***normal order***. See examples to be clear about the order.

All records should be processed sequentially from beginning to end.

3. Program and input specification

The main C++ program will become the executable to be tested by the TAs. The Result file should be written to another text file (output file), provided with the Command line. Notice the input and output files are specified in the command line, not inside the C++ code. Notice also the quotes in the program call, to avoid Unix/Windows gets confused.

Assumptions:

- The input file is a small plain text file; no need to handle binary files.
- Each input file may contain maximum 1000 line, if you want to implement stack and queue using array.
- Note that, input and output, all are case sensitive.
- The output should be exactly matched with the format.

The general call to the executable is as follows:
css “input=input1.txt;output=output1.txt”

You can call the program with another command line type,
css input=input1.txt output=output1.txt

4. Examples

Example 1 of input and output,
Input1.txt

```
buy Book_A          // insert Book_A into your Queue and Stack
buy Book_B
buy Book_C
buy Book_D
```

buy Book_E

sale Book_A // Remove Book_A from your Queue and Stack, and find out the cost for stack and queue
sale Book_D
sale Book_A
sale Book_E

Command line:

css input=input1.txt output=output1.txt

output1.txt

Book_A finding cost at stack: 5, at queue: 1
Book_D finding cost at stack: 3, at queue: 3
Book_A not found
Book_E finding cost at stack: 1, at queue: 1

Example 2 of input and output,

Input2.txt

buy Algorithms
sale Algorithms
sale Algorithms
buy Data Structures
buy Data Structures
buy Data Structures
buy Programming in C
sale Data Structures

Command line:

css input=input2.txt output=output2.txt

output2.txt

Algorithms finding cost at stack: 1, at queue: 1
Algorithms not found
Data Structures finding cost at stack: 2, at queue: 1

Example 3 of input and output,

Input3.txt

buy Mobile Computing
buy Web Security
buy Data Mining
buy Databases
buy Algorithms
buy Structured Programming
sale Intrusion Detection
sale Web Security
sale Network Security
sale Mobile Computing
sale Databases

Command line:

css input=input3.txt output=output3.txt

Output3.txt

Intrusion Detection not found
Web Security finding cost at stack: 2, at queue: 2
Network Security not found
Mobile Computing finding cost at stack: 5, at queue: 5
Databases finding cost at stack: 2, at queue: 2

Explanation for 1st case:

After adding all those five books into **stack**,

1	Book_E
2	Book_D
3	Book_C
4	Book_B
5	***Book_A***

After Selling Book_A, the stack will be,

1	Book_B
2	Book_C
3	***Book_D***
4	Book_E

After selling Book_D,

1	Book_C
2	Book_B
3	Book_E

And after the unsuccessful search, the whole stack will reverse,

1	***Book_E***
2	Book_B
3	Book_C

After adding all these 5 books into **queue**,

Book_A	Book_B	Book_C	Book_D	Book_E

After selling Book_A,

Book_B	Book_C	***Book_D***	Book_E

After selling Book_D,

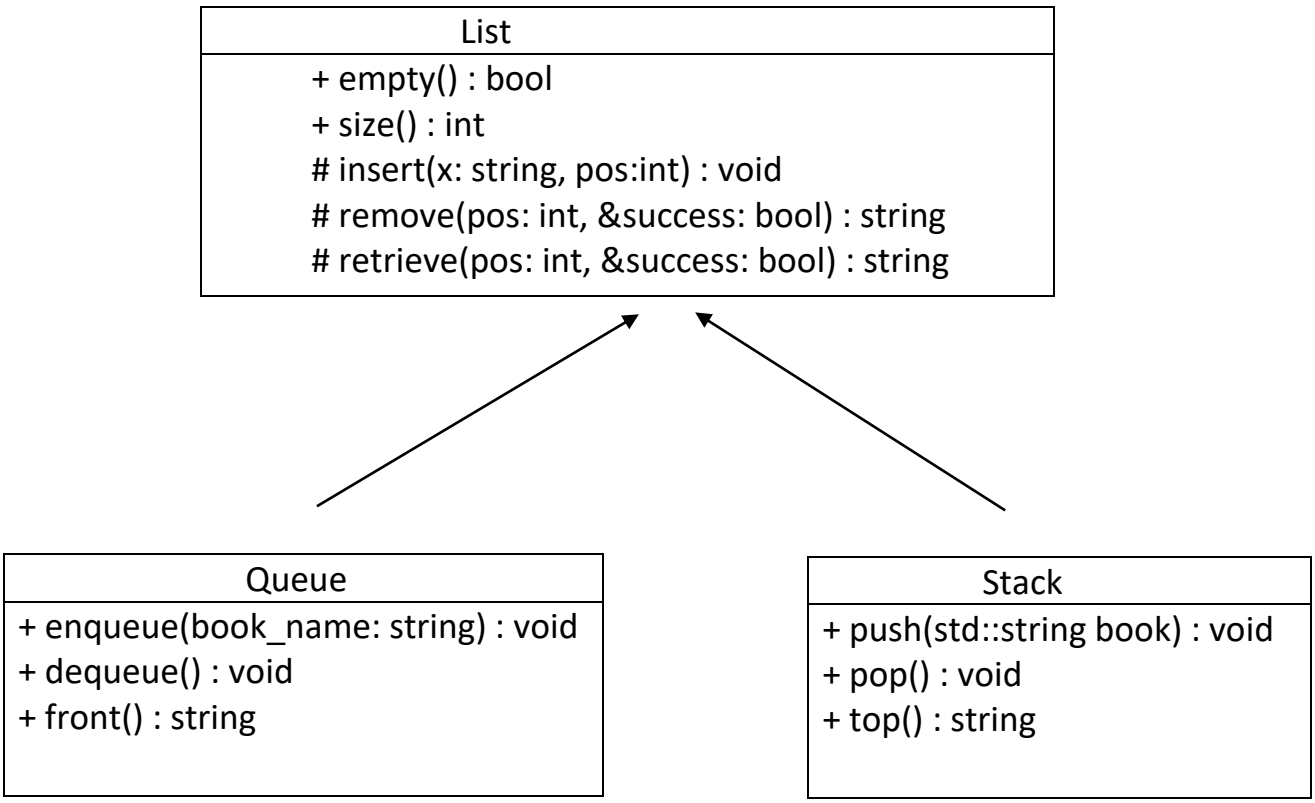
Book_E	Book_B	Book_C

After unsuccessful search, it will be same

Book_E	Book_B	Book_C

5. Implementation Specification:

This homework is mainly for the manipulation of object-oriented programming behavior, e.g. abstraction, encapsulation, inheritance, and polymorphism. You have to implement Stack and Queue by yourself. The basic idea is that you will implement a base class for List, can be either Linked list or array list. Then you will derive the List class to implement your Stack and Queue class. Your Stack and Queue class will have the same base, List, class. Here is the UML based class diagram,



6. Requirements

- Homework is an individual. Your homework will be automatically screened for code plagiarism against code from the other students and code from external sources. If you copy/download source code from the Internet or a book, it is better for you to acknowledge it in your comments, instead of the TAs detecting it. Code that is detected to be copied from another student (for instance, renaming variables, changing for and while loops,

changing indentation, etc) will result in ”Fail” in the course and being reported to UH upper administration.

- timeout is set to 2s.

7. Hand over your homework

- Homework 6 need to be handed over to our Linux server, follow the link here http://www2.cs.uh.edu/~rizk/homework.html.

- Make sure to create a folder under your root directory, name it hw6 (name need to be lower case), only copy your code to this folder, no test case or other files needed. If you use ArgumentManager.h, don’t forget to hand over it too.

PS. This document may have typos, if you think something illogical, please email TAs or Teacher for confirmation.