CSC230 Lab 8

Due: April. 10th, 11:59pm

Goal: This lab is split up into 2 parts, each of which will give you practice with queues, stacks, and algorithmic design:

1. MyQueue.h: Implement a queue using two stacks
2. MyStack.h: Implement a stack using two queues

There are two files: stackT.cpp and queueT.cpp. These two files use stack and queue defined by C++ STL. Read these two files, understand how stack and queue work, then work on your own files. After you finish implementing MyQueue.h and MyStack.h, use MyQueueTest.cpp and MyStackTest.cpp to test that your new stack and queue implementations work properly.

You do not have to implement all functions of stack and queue STL.

http://www.cplusplus.com/reference/stack/stack/
http://www.cplusplus.com/reference/queue/queue/

Just implement push(), pop(), top() for MyStack class; push(), pop(), and front() for MyQueue class.

In this lab, you MUST use template to implement both classes.

Part 1: MyQueue

MyQueue implementation

Implement the stubbed out functions in the MyQueue class using two stacks. The function specifications are shown below. The MyQueue class has 2 stacks as instance variables, which represent the inner state of the queue. Therefore, if an object has been dequeued from the queue it should not be in either stack. If an object is enqueued into the queue it should be present in at least one stack.

Methods to implement (specs also in the .h file):

```
#include <iostream>
#include <stack>
using namespace std;

template <class T>
class MyQueue {
   // these two stck are instance variables
   // by default, the access is private
   stack<T> first;
```

```
    STACK<T> SECOND;

    PUBLIC:

    // RETURN THE VALUE OF THE OLDEST MEMBER
    T FRONT(){
      // PLEASE IMPLEMENT THIS METHOD
    }

    // ADD VALUE VAL TO MYQUEUE
    VOID PUSH(T VAL){
      // PLEASE IMPLEMENT THIS METHOD
    }


    // REMOVE THE OLDEST MEMBER FROM MYQUEUE
    VOID POP(){
      // PLEASE IMPLEMENT THIS METHOD
    }
};
```

Use queueTest.cpp to test your implementation of the MyQueue class.

Part 2: MyStack

MyStack implementation

Implement the stubbed out functions in the MyStack class using two queues. The MyStack class has 2 queues as instance variables, which represent the inner state of the stack. So, like with MyQueue, if an item is pushed onto the stack, it should be in at least one queue, and if an object is popped, it should not be in either queue.

```
#INCLUDE <IOSTREAM>
#INCLUDE <QUEUE>
USING NAMESPACE STD;

TEMPLATE <CLASS T>
CLASS MYSTACK {

 // DEFINE TWO INSTANCE VARIABLES
 // BY DEFAULT, THEY ARE PRIVATE
 QUEUE <T> FIRST;
 QUEUE <T> SECOND;

 PUBLIC:

 // RETURN THE LATEST VALUE OF MYSTACK
 T TOP(){
   // PLEASE IMPLEMENT THIS METHOD
 }
```

```
  // ADD VALUE VAL TO MYSTACK
  VOID PUSH(T VAL){
    // PLEASE IMPLEMENT THIS METHOD
  }

  // REMOVE THE OLDEST VALUE FROM MYSTACK
  VOID POP(){
    // PLEASE IMPLEMENT THIS METHOD
  }
};
```

Use stackTest.cpp to test your implementation of the MyStack class.

Hint:
Both stack class and queue class from C++ STL have a method called empty(), which checks whether the stack object/queue object is empty or not. You will need this method when you implement MyQueue class and MyStack class.

Wrap up

Submit two .h files to Canvas