Lab Fa

Goals-
Implement linear data structures using linked structures

You will create two abstract data types, a stack and a queue. You will not use any existing containers, such as the STL. You will use a linked linear structure to store the values and provide the necessary functions to interact with that linked structure. For our purposes the data for both is an int!! You will not have one function that both removes a node and gives you the value in that node. You will also have a function to test if the structure is empty.

You will use appropriate pointer variables to the element/node that is at the beginning of the structure, the end of the structure, or both. You will not use a pointer to iterate or go to other individual elements of the structure. In both cases removing a node is restricted to just one end; either the first or the last.

Each data structure will be in a separate source file, with the declarations for each in a separate header file. Your driver function/program will be in a separate source file. Those declarations can be in a separate header file if you choose.

STACK

The stack is a first in last out structure. You add to the top and can only look at or take off the top. You only need a singly linked list to implement it. Why is that? You will implement these functions, with appropriate parameters:

```
void push()  // puts on item onto the structure
int peek()   // returns the value on top of the structure
void pop()   // removes the top item in the structure
isEmpty()
```

QUEUE

The queue is a first in first out structure. You add to the back and can only look at or take off the front. For the queue sit down with pencil and paper and study the required pointer manipulations. You only have the link(s) in each node, a pointer to the front and a pointer to the back. You can use a singly linked structure if you want. Have your pointer algorithms written out before you start writing your code! You will implement these functions, with appropriate parameters:

```
void addBack()      // puts on item at the end of the structure
int getFront()      // returns the value at the front of the structure
void removeFront()  // removes the first item in the structure
isEmpty():
```

TESTING

You must also write a driver program that uses your stack and queue to demonstrate they work correctly. You should prompt the user to enter a series of integers, with some way to indicate they have finished entering values. Just enter the each number into your stack and queue. Then you print out the values as you remove them from your structure. Make sure you label the output as to which is coming from the stack or the queue. How can you tell they are working correctly? Explain in the testing document.

Your code should also generate an error if a user attempts to remove more items than were entered into the structure. Add to your driver program a test to fill the queue and stack with N values and attempt to print N+1. What happens? Why do you not need to test for putting too many items into either the stack or queue?

What to submit-

You will submit the following files to TEACH-

Code to implement your stack, both header and source files

Code to implement your queue, both header and source files

Code to demonstrate the operation of your stack and queue.

PDF file with test results and answers to questions.

HINT: Please do the stack first. The pointer manipulation is easier and more obvious. Then use that experience to develop the pointer algorithms for the queue and then write that code.

Grading

Programming style- 1 point

Code to implement your stack
header file- 1 point
source file- 2 points

Code to implement your queue
header file- 1 point
source file- 2 points

Code to test your stack and queue- 2 point

Test results and answers- 1 point