Online 2 (November 17, 2018)

Time: 40 Minutes Marks: 10

Introduction

Queue is a linear data structure that maintains FIFO (First In First Out) ordering. A queue can be implemented using an array or a linked list. However, for this problem, you have to implement a queue using **two stacks**. You need to use your offline on stack for this purpose.

A queue can have different functionalities like inserting an element, removing the least recently inserted element, reporting number of elements in the queue and so on (details in the next section). However, your implementation must follow an object oriented approach.

Problem Specification

You have to design & implement a class named **queue** having only **two** variables (object or pointer) of **stack** type. **You cannot use any other variable**. The operations to be implemented are described in brief.

- **Initialization:** Write suitable constructor(s) to initialize two stacks.
- **Enqueue:** Insert an integer or a stack object into your queue. The example below illustrates each of these.

Current elements in Queue	Enqueue	Elements after Enqueue
2, 4	10	2, 4, 10
2, 4	Stack with elements: 5, 6	2, 4, 6, 5

  Write suitable functions to achieve this. Note that, name of the functions must be the same for both these enqueue operations.

- **Dequeue:** Remove the least recently inserted element from queue. Note that, this function will always return an integer. For example, let the elements of a queue be {2, 4} [enqueue 2, then enqueue 4]. Now dequeue operation will return 2 & remove it from the queue. Next dequeue will return 4 after removing it from the queue.
- **Size:** Reports number of elements currently in the queue.
- **Convert:** Returns a stack with the same elements. For example, if queue contains {2, 4} [enqueue 2, then enqueue 4], the stack to be returned will have {2, 4} [push 2, followed by push 4]. But to get the queue elements **only enqueue & dequeue** operations can be used. And the **queue must be unchanged** afterwards (for instance, if you only dequeue the elements & push into the stack, the queue will be empty).

Input Output Specification

In your main function, you will declare a queue object. You have to prompt for input in a loop. Show a menu like the one shown below.

1: Enqueue an int
2: Enqueue a stack
3: Dequeue
4: Size
5: Convert
6: Exit

Based on the response, take necessary inputs next. For example, if the user chooses 2, take *n*+1 integers as input where *n* denotes the size of stack followed by *n* elements of the stack.

If the user chooses 5, pop & print all the elements of the stack in the main function (after you have returned it).

Finally, when the user chooses 6, dequeue all the elements of your queue object one by one & print them separated by space.

Marks Distribution:

Task	Initialization	Enqueue	Dequeue	Size	Convert	Total
Marks	2	2	3	1	2	10

Hints:

Suppose the two stacks in the queue are called s1 & s2. For enqueue, you can simply push in s1. If you do so, then for dequeue, you can do the following:

```
If (s2 is empty)
        push all the elements of s1 into s2 until s1 becomes empty.
End If
Pop an element from s2.
```

Special Instructions:

* In order to get full marks,

- Your queue class must not have any additional variable.

- You need to implement dequeue operation with the minimum possible number of function calls.

* If required, you can modify the stack you implemented in offline 2. But basic properties like inserting elements to & deleting elements from the top, no provision for random access into the stack etc. must be maintained.

* If you **have not completed Offline 2**, you can use built-in **stack** data structure **(not queue)** from standard template library of C++ (if you can use it), to get partial marks based on what you can implement in this online.