National University of Computer and Emerging Sciences

Lab Manual

for

Data Structure

Course Instructor	Dr. Zareen Alamgir
Lab Instructor(s)	Ms. Mamoona Akbar Ms. Ammara Nasir
Section	DS BCS-3C
Semester	FALL 2022

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 05

Objectives:

After performing this lab, students shall be able to revise:

- ✔ Circular Queue using Array
- ✔ Stack using LinkList

Problem 1

A circular queue is the extended version of a regular queue where the last element is connected to the first element. Thus, forming a circle-like structure. In this lab you have to implement CircularQueue class with a dynamic array as data member and following
member functions:

- **T front()** : Returns the first item of Queue.
- **T rear():** Returns last item of the Queue.
- **void enqueue(T e):** Inserts element in the Queue. It also checks if queue is full before inserting an element. Element should be added in the position pointed by rear.
- **T dequeue():** Delete element from Queue and returns it. It should check if queue is empty or not. If not empty then return the value pointed by front.
- **int size():** Returns size of the Queue.
- **bool Isfull():** Returns true if queue is full otherwise false.
- **bool Isempty():** Returns true if queue is empty.
- **Void print():** Display the
- **CircularQueue():** Default constructor and create queue of size 5
- **CircularQueue(int s):** Default constructor and create queue of size s
- **~CircularQueue() :** destructor

Driver/main Program

Show the final contents of the array after the following code is executed:

```
for (int k = 1; k <= 7; k++)
        Q.enqueue(k);
```

```
for (int k = 1; k <= 4; k++)
{
        Q.enqueue(Q.dequeue());
        Q.dequeue();
}
```

Final answer:

0	1	2	3	4	5	6
	3	5	7			

Problem 2

1) Implement a template Stack class using Link List with one top pointer only.

- **bool Push (T val)** // Add an element in Stack. Returns False if push operation is unsuccessful otherwise True.
- **bool Pop ()** // Removes top element from Stack. Returns true if operation is successful otherwise false if stack is empty with some error message.
- **bool Top(T&)** // returns, but does not delete, the topmost element from the stack via the parameter passed by reference. It returns false via a return statement if there is no element in the stack, else it returns true and assigns the top most element to the parameter passed by reference.
- **bool IsEmpty()**
- **bool IsFull()**
- **Stack()** //default constructor. Creates a stack of default size 10
- **Stack(int size**) // Parameterized Constructor. Creates a stack of size = size
- **~Stack()** // Destructor

Driver/main Program

As you have implemented stack and queue now perform following operations. Given a 5 element queue Q (from front to rear: 1, 3, 5, 7, 9), and an empty stack S, remove the elements one-by-one from Q and insert them into S, then remove them one-by-one from S and re-insert them into Q.

The queue now looks like (from front to rear)

9, 7, 5, 3, 1

2). Write a function addlargenumbers() which takes two very large numbers and returns sum of these numbers. Here are a few instructions that you need to keep in mind:

- Use the **Stack** class (for storing integers) that you have coded above.
- Your program should read two numbers from console and store them as two strings.Assume that the maximum number of numerals (digits) in a number is 30. ***Read the numerals (digits) of the first number and store the numbers corresponding to them on stack. And read the numerals (digits) of the second number and store the numbers corresponding to them on stack2***

- After reading input, your program should determine the sum of these two large numbers as discussed in class and store this result in Resultstack.

  Stack1: **23456789123456789123456789**
  Stack2: **123454321123454321123454321**
  Resultstack: **246911110246911110246911110**

GOOD LUCK