**Data Structures & Algorithms LAB**

(BSCS-F18 Morning & Afternoon)

**Lab # 6**

**Task # 1**

**Queue class is given below**

Class Queue

{

private :

int \*q;

int maxsize, currentsize;

int back , front;

// constructor

//destructor

bool enque(int val)

bool dequeue(int &val)

bool isfull()

bool is empty()

}

**Implement following function of Queue.**

// constructor

//destructor

bool enque(int val)

bool dequeue(int &val)

bool isfull()

bool is empty()

Use a driver program that will check operations of Queue.

**Task # 2**

Implement assignment operator for Queue

**Task # 3**

Implement a function that returns true if two queues are identical and false otherwise.

You can only use queue operations to implement it.

**Task # 4**

**Indicate whether each of the following application would be suitable for a queue or a stack (or none of them). Write your answer in two lines.**

1. A company wants to evaluate employee record so as to layoff some workers on the basis of service time (the most recently hired employees are laid off first)
2. A program is to keep track of patients as they check into a clinic, assigning them to doctors on a first come first served basis.
3. A program to solve a maze is to backtrack to an earlier position (the last place where a choice was made) when a dead end position is reached. An inventory of parts is to be processed by parts number
4. An operating system is to process requests for computer resources by allocating the resources in the order in which they are requested
5. A grocery chain wants to run a simulation to see how the average customers wait times affected by changing the number of checkout lines in their store.
6. A dictionary of words used by a spelling checker is to be initialized
7. Customers are to take number at a bakery and be served in order when their number comes up.
8. Gamblers are to take numbers in the lottery and win if their numbers are picked.

**Task # 5**

As we have seenin task 1, in Queue data structure, elements are inserted at the back

and removed from the front end of the queue. A **Double Ended Queue** (Deque) is a queue in which you can insert and remove elements **both at the *front* and at the *back* of the queue**. In this task, you are required to implement a class **Deque** (which will be used to store integers). The **Deque** class should have the following public member functions:

**Deque (int n)** Constructor to create a Deque which can contain up to **n** elements.

**~Deque ()** Destructor to destroy the Deque.

**bool isEmpty ()** returns true if the Deque is empty, and false otherwise.

**bool isFull ()** returns true if the Deque is full, and false otherwise.

**void display ()** to display the elements of Deque **from front to back**.

**void makeEmpty ()** to make the Deque empty.

**bool insertAtFront (int val)** to insert **val** at the front of the queue.

**bool insertAtBack (int val)** to insert **val** at the back of the queue.

**bool removeFromFront (int& val)** to remove an integer from the front of the

queue and store it in the variable **val**.

**bool removeFromBack (int& val)** to remove an integer from the back of the

queue and store it in the variable **val**.

The last four functions should have the **bool** return type. These functions should return **true** if the desired operation was performed successfully. Otherwise these functions should return **false**.

Note that:

• The **time complexity** of **last 4 member functions** should be **constant i.e. *O*(1)**. Also note

that there will **NOT** be any input / output statement in these 4 functions.

• Your Deque class should refuse the insertion of a new element ONLY when there are no

empty slots left in the Deque. So, make sure that all the boundary cases are properly handled.

You are also required to write a menu-driven program which allows the user to use all of the

aforementioned functionalities of the **Deque** class. A sample menu is shown below (Text shown

in **Red** is entered by the user):

**Enter the size of Deque: 10**

**1. Insert value at Front**

**2. Insert value at Back**

**3. Remove a value from Front**

**4. Remove a value from Back**

**5. Display the Deque**

**6. Make the Deque Empty**

**7. Exit**

**Task # 6**

1. Write down the logic you used for task 5.
2. Provide txt file rather than word file for this answer
3. Provide you answer of task 4 in the txt file
4. Done your task # 4 and Task #6 in the same txt file
5. Submit only one txt file.

**Task # 7**

**Only Practice Required:**

Implement a function that convert infix expression to postfix by using stack.