Chandigarh College of Engineering and Technology, (Degree Wing) Chandigarh
Department of Computer Science and Engineering

CS-301: Data Structures   Assignment: 07   Date: 03.09.2024

S. No.	**Note:** Start answer of a fresh question from fresh page only. Direct answer to a question will not be entertained.	course outcome (CO)
1.	Design an algorithm for a input restricted **deque**. Implement the designed algorithm.	CO4
2.	Design an algorithm for a output restricted **deque.** Implement the designed algorithm.	CO4
3.	In the following Deletion and Insertion Algorithms From/Into original Queue is **Q** and size is n (size/capacity). The **Info** is used to receive the deleted information. There two pointer rear and front used in the Q. these pointers used to insert and delete an item/inform from/into queue Q. Implement these algorithms in C/C++ using data structures Array.	CO4

```
Delete_Deque(Q, rear, front)
{
        Info: Parameter to backup the information going to be deleted
        Step 1: [Check for underflow]
                If (front = 0 and rear = 0)
                Output "Underflow" and Return
        Step 2: [Backup the element at front end]
                If (front >0)
                        info ← Q [front]
        Step 3: [Check queue for empty]
                If (front = rear ){
                        front ← 0
                        rear← 0
                }Else
                        front ← front + 1
        Step 4: [Backup the element at the rear end]
                If (rear > 0)
                        Info ←Q [ rear]
        Step 5: [Check queue for empty]
                If (front = rear ){
                        front←0
                        rear ←0
                }Else
                        rear ← rear – 1
        Step 6: Return(info)
}

Insertion_Insert(Q, front, rear, info)
{
        Step 1: [Check overflow condition]
                If ((rear = n) and (front = 1))
```

```
            Output “Overflow” and Return (0)
        Step 2: [Insert element at the front end]
            If (front > 0)
            {
                front ← front – 1
                Q [front] ←info
                Return (1)
            }
        Step 3: [Insert element at the rear end]
             If (rear < n)
            {
                Rear ← rear + 1
                Q [rear]  ← info
                Return (1)
            }
        Step 4: [End]
            Return (0)
    }
```