Assignment - 4

Q1. Write pseudocode / C code for Linear Queue.

```
#include <stdio.h>
#define MAX 5

typedef struct queue
{
int front ;
int rear ;
intele[MAX] ;
}Queue;

//Intialze Queue
void init( Queue *q)
{
q->rear = -1;
q->front = 0;
}
int isFull(Queue *q)
{
int full=0;
if(q->rear== MAX-1)
full = 1;
return full;
}

//To check Queue is empty or not
int isEmpty(Queue *q)
{
int empty=0;
if( q->front == q->rear+l)
empty = 1;
return empty;
}

//Insert item into queue
void insertQueue(Queue *q,int item)
{
if( isFull(q))
{
printf("\nQueue Overflow");
return;
```

```
}
q->ele[++(q->rear)] = item;
printf("\nlnserted item: %d",item);
}

//Delete item from queue
int deleteQueue( Queue *q, int * item)
{
if( isEmpty(q))
{
printf("\nQueue Underflow");
return -1;
}
*item = q->ele[(q->front)++];
return 0;
}

int main()
{
int item = 0;
Queue q;
init(&q);
insertQueue(&q,10);
insertQueue(&q,20);
insertQueue(&q,30);
insertQueue(&q,40);
insertQueue(&q, 50);
insertQueue(&q,60);
if( deleteQueue( &q, &item ) == 0 )
printf("\nDeleted item: %d",item);
if( deleteQueue( &q, &item ) == 0 )
printf("\nDeleted item: %d",item);
if( deleteQueue( &q, &item ) == 0 )
printf("\nDeleted item: %d",item);
if( deleteQueue( &q, &item ) == 0 )
printf("\nDeleted item: %d",item);
if( deleteQueue( &q, &item ) == 0 )
printf("\nDeleted item : %d",item);
if( deleteQueue( &q, &item ) == 0 )
printf("\nDeleted item : %d",item);
printf(”\n");
return 0;
```

Q2. Write pseudocode / C code for Doubly Ended Queue.

Algorithm for Insertion at rear end

```
Step-1:  [Check for over ow]
                if(rear==MAX)
                        Print("Queue is Over ow”);
                        return;
Step-2:   [Insert Element]
                else
                        rear=rear+1;
                        q [rear]=no;
                 [Set rear and front pointer]
                if rear=0
                        rear=1;
                if front=0
                        front=1;
Step-3: return
```

Algorithm for Insertion at front end

```
Step-1  :   [Check for the front position]
                if(front<=1)
                        Print("Cannot add item at the front”);
                        return;
Step-2  :   [Insert at front]
                else
                        front=front-1;
                        q [front]=no;
Step-3   : Return
```

Algorithm for Deletion from front end

```
Step-1  [ Check for front pointer]
                if front=0
                        print(" Queue is Under ow”);
                        return;
Step-2  [Perform deletion]
                    else
                       no=q [front];
                       print(“Deleted element is”,no);
               if front=rear
                        front=0;
                        rear=0;
               else
                        front=front+1;
Step-3: Return
```

Algorithm for Deletion from rear end

```
Step-1:  [Check for the rear pointer]
               if rear=0
                        print(“Cannot delete value at rear end”);
                        return;
Step-2:   [Perform deletion]
               else
                        no=q [rear];
               if front= rear
                        front=0;
                        rear=0;
               else
                        rear=rear-1;
                        print(“Deleted element is”,no);
Step-3: Return
```