Lab 3: Implementation of linear queue as an ADT.

Theory:

A data structure of ordered items such that items can be inserted only at one end and removed at the other end is called queue. Placing an item in a queue is called "insertion or enqueue", which is done at the end of the queue called "rear".Removing an item from a queue is called "deletion or dequeue", which is done at the other end of the queue called "front".

Let, MAX be the maximum size of queue, Size be the current size of queue, front be the front of the queue, rear be the rear of the queue, Queue[MAX] be the array representing queue.

Algorithm to enqueue:

1. Start
2. Initialize front=0 and rear=-1
3. If rear = MAX-1 then display queue is full.
4. Else, Ask for data
   rear = rear + 1
   Queue[rear] = data
5. Stop

Algorithm to dequeue:

1. Start
2. If rear<front display queue is empty.
3. Else, data= queue[front]
   return data
   front = front +1
4. Stop

Source Code:

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#define MAX 10
struct LQ
{
    int front;
    int rear;
    int queue[MAX];
};
void enqueue(struct LQ*,int);
int dequeue(struct LQ*);
int check_front(struct LQ*);
int check_rear(struct LQ*);
int main()
{
    struct LQ q={0,-1};
    int choice,data;
    do
    {
        printf("\n");
        printf("Select an option. \n");
        printf("1. ENQUEUE\n");
        printf("2. DEQUEUE\n");
        printf("3. CHECK FRONT\n");
        printf("4. CHECK REAR\n");
        printf("5. EXIT.\n");
        printf(">");
        scanf("%d",&choice);
        switch (choice)
        {
            case 1:
                printf("Enter data : ");
                scanf("%d",&data);
                enqueue(&q,data);
                break;
            case 2:
                if(q.rear<q.front)
                    printf("QUEUE IS EMPTY \n");
                else
                    printf("Removed element is : %d \n",dequeue(&q));
                break;
            case 3:
                if(q.rear<q.front)
```

```
                    printf("QUEUE IS EMPTY \n");
                else
                    printf("Data : %d \n",check_front(&q));
                break;
            case 4:
                if(q.rear<q.front)
                    printf("QUEUE IS EMPTY \n");
                else
                    printf("Data : %d \n",check_rear(&q));
                break;
        }
        }while (choice!= 5);
        return 0;
}

void enqueue(struct LQ *Q, int a)
{
    if(Q->rear==MAX-1)
        printf("QUEUE IS FULL \n");
    else
    {
        Q->rear=Q->rear+1;
        Q->queue[Q->rear]=a;
        printf("%d was added into the queue\n",a);
    }
}
int dequeue(struct LQ *Q)
{
    int ret;
    ret=Q->queue[Q->front];
    Q->front=Q->front+1;
    return ret;
}
int check_front(struct LQ *Q)
{
    return (Q->queue[Q->front]);
}
int check_rear(struct LQ *Q)
{
    return (Q->queue[Q->rear]);
}
```

Output:

```
Select an option.
1. ENQUEUE
2. DEQUEUE
3. CHECK FRONT
4. CHECK REAR
5. EXIT.
>1
Enter data : 11
11 was added into the queue
```

Conclusion: In this lab we implemented linear queue as an ADT in C programming language.

.