Lab 4: Implementation of circular queue as an ADT.

Theory: The linear queue is of a fixed size. Array implementation of linear queues leads to the queue is full state even though the queue is not actually full. When a new item is inserted at the rear, the arrow to rear moves downwards. Similarly, when an item is deleted from the queue the front arrow moves downwards. After a few insert and delete operations the rear might reach the end of the queue and no more items can be inserted although the items from the front of the queue have been deleted and there is space in the queue.

To solve this problem, queues implement wrapping around. Such queues are called Circular Queues.

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

Algorithm for Enqueue:

```
1. Start
2. If SIZE == MAX then display Queue is full.
3. Else, Read data.
            rear = (rear + 1) % MAX
            Queue[rear] = data
            SIZE=SIZE+1
4. Stop
```

Algorithm to Dequeue:

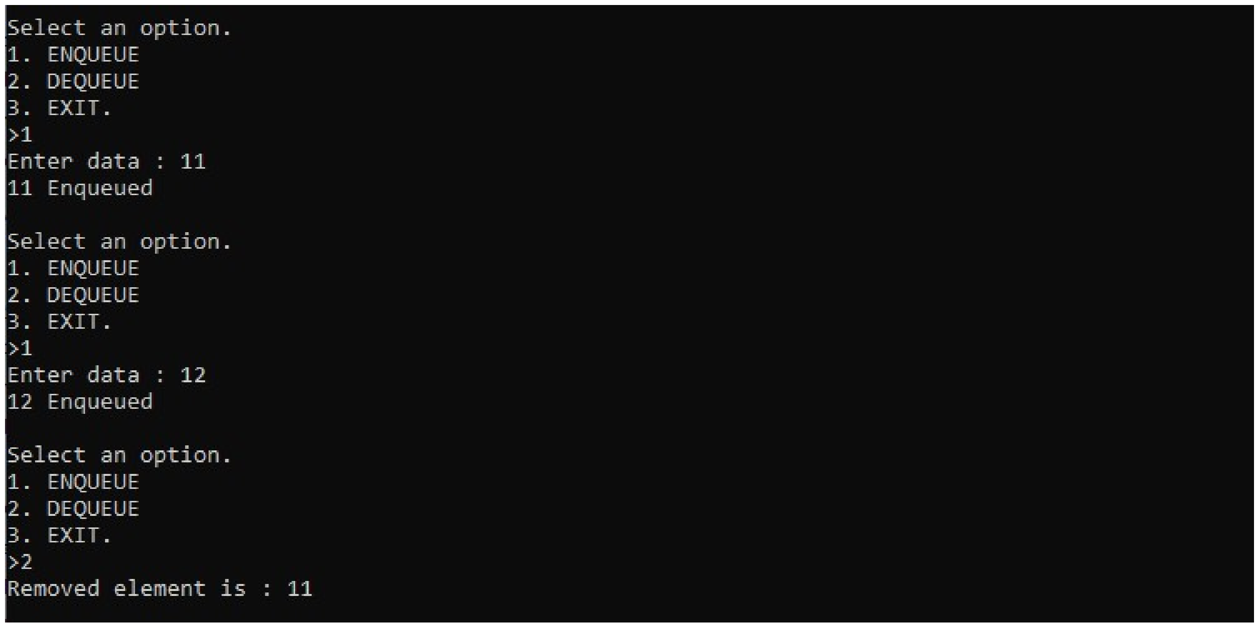
```
1. Start
2. If SIZE == 0 then display Queue is empty.
3. Else
            data = Queue[front]
            front = (front + 1) % MAX
            SIZE = SIZE-1
4. Stop
```

Source Code:

```c
#include <stdio.h>
#define MAX 5
struct CQ
{
    int size, front, rear, queue[MAX];
};
void enqueue(struct CQ*,int);
int dequeue(struct CQ*);
int main()
{
    struct CQ q={0, 0, -1};
    int choice,data;
    do
    {
        printf("\n");
        printf("Select an option. \n");
        printf("1. ENQUEUE\n");
        printf("2. DEQUEUE\n");
        printf("3. EXIT.\n");
        printf(">");
        scanf("%d",&choice);
        switch (choice)
        {
            case 1:
                if(q.size == MAX)
                    printf("QUEUUE IS FULL\n");
                else
                    {
                        printf("Enter data : ");
                        scanf("%d",&data);
                        enqueue(&q, data);
                        printf("%d Enqueued\n",data);
                    }
                break;
            case 2:
                if(q.size == 0)
                    printf("QUEUE IS EMPTY\n");
                else
                    printf("Removed element is : %d \n",dequeue(&q));
                break;
            case 3:
                printf("BYE\n");
                break;
        }
        }while (choice!= 3);
        return 0;
}

void enqueue(struct CQ *Q, int a)
{
        Q->rear = (Q->rear+1)%MAX;
        Q->queue[Q->rear]=a;
        Q->size += 1;
}
int dequeue(struct CQ *Q)
{
    int ret = Q->queue[Q->front];
    Q->front=(Q->front+1)%MAX;
    Q->size -= 1;
    return ret;
}
```

Output:



```
Select an option.
1. ENQUEUE
2. DEQUEUE
3. EXIT.
>1
Enter data : 11
11 Enqueued

Select an option.
1. ENQUEUE
2. DEQUEUE
3. EXIT.
>1
Enter data : 12
12 Enqueued

Select an option.
1. ENQUEUE
2. DEQUEUE
3. EXIT.
>2
Removed element is : 11
```

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