Program

```
#include <stdio.h>
#define MAX_SIZE 5
int queue[MAX_SIZE-1];
int rear=-1;
int front=-1;
void enqueuefront(int val){
    if((front==0&&rear==MAX_SIZE-1)||front==rear+1)
      printf("Queue Overflow\n");
    else {
      if(front==-1&&rear==-1){
      front=rear=0;
      queue[front]=val;}
      else if(front ==0){
      front=MAX_SIZE-1;
      queue[front]=val;}
      else
      front --;
      queue[front]=val;}}
void enqueuerear(int val){
    if((front==0&&rear==MAX_SIZE-1)||front==rear+1)
      printf("Queue Overflow\n");
    else {
      if(front==-1&&rear==-1){
      front=rear=0;
      queue[rear]=val;}
      else if(rear==MAX_SIZE-1){
      rear=0;
      queue[rear]=val;}
      else
      rear++;
      queue[rear]=val;}}
void dequeuefront(){
    if(front==-1&&rear==-1)
      printf("Queue Underflow\n");
    else if(front==rear){
       int val=queue[front];
       rear=front=-1;
      printf("Value %d has been dequeued\n",val);}
    else if(front==MAX_SIZE-1){
       int val=queue[front];
       front=0;
      printf("Value %d has been dequeued\n",val);}
    else{
      int val=queue[front];
      front++;
      printf("Value %d has been dequeued\n",val);
      }}
void dequeuerear(){
    if(front==-1&&rear==-1)
      printf("Queue Underflow\n");
    else if(front==rear){
       int val=queue[rear];
       rear=front=-1;
      printf("Value %d has been dequeued\n",val);}
    else if(rear==0){
       int val=queue[rear];
       rear=MAX_SIZE-1;
      printf("Value %d has been dequeued\n",val);}
    else{
      int val=queue[rear];
```

Experiment No: 10
Date:

DOUBLE ENDED QUEUE

Aim:

To implement Double Ended Queue.

Algorithm:

1. Start
2. Define MAXSIZE as 5 and declare queue of size MAXSIZE - 1.
3. Initialize rear and front to -1 (empty queue).
4. Function enqueuefront(val):

```
If (front == 0 && rear == MAX_SIZE - 1) || (front == rear + 1):
       Print "Overflow"
Else if front == -1 && rear == -1:
       Set front = 0, rear = 0,
      queue[front] = val
Else if front == 0:
       Set front = MAX_SIZE - 1,
       queue[front] = val
Else:
     Decrement front,
     queue[front] = val
```

5. Function enqueuerear(val):

```
If (front == 0 && rear == MAX_SIZE - 1) || (front == rear + 1):
      Print "Overflow"
Else if front == -1 && rear == -1:
      Set front = 0, rear = 0,
      queue[rear] = val
Else if rear == MAX_SIZE - 1:
     Set rear = 0, queue[rear] = val
Else:
     Increment rear,
    queue[rear] = val
```

6. Function dequeuefront():

```c
      rear--;
      printf("Value %d has been dequeued\n",val);
      }}
void display() {
    int i;
    if (front == -1) {
        printf("Queue is Empty\n");

    }
    else{

      for(i=front;i!=rear;(i=(i+1)%MAX_SIZE)){
       printf("%d\t",queue[i]);}
       printf("%d\t",queue[i]);}}
int main(){
  int choice,value;
  while (1) {
        printf("\nQueue Operations:\n");
  printf("1-->ENQUEUE F\n2-->ENQUEUE R\n3-->DEQUEUE F\n4-->DEQUEUE R\n5-->DISPLAY\n6-->EXIT\n");
  printf("Enter choice:");
  scanf("%d",&choice);
  switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueuefront(value);
                break;

            case 2:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueuerear(value);
                break;

            case 3:
                dequeuefront();
                break;

            case 4:
                dequeuerear();
                break;

            case 5:
                display();
                break;

            case 6:
                printf("Exiting...\n");
                return 0;
                break;

            default:
                printf("Invalid choice!\n");
                break;
        }
    }return 0;}
```

```
If front == -1 && rear == -1:
      Print "Underflow"
Else if front == rear:
      Print queue[front],
      set front = -1, rear = -1
Else if front == MAX_SIZE - 1:
      Print queue[front],
     set front = 0
Else:
     Print queue[front],
     increment front
```

7. Function dequeuerear():

```
If front == -1 && rear == -1:
      Print "Underflow"
Else if front == rear:
      Print queue[rear],
      set front = -1, rear = -1
Else if rear == 0:
     Print queue[rear],
    set rear = MAX_SIZE - 1
Else:
    Print queue[rear],
    decrement rear
```

8. Create function Display()

```
If front == -1:
   Print "Queue is Empty"
Else:
   Set i = front
    While i != rear:
            Print queue[i],
            set i = (i + 1) % MAX_SIZE
    Print queue[rear]
```

9. Create function main()

```
Repeat:
   Print menu
   Read choice
If choice == 1:
          Call enqueuefront(val)
If choice == 2:
          Call enqueuerear(val)
If choice == 3:
          Call dequeuefront()
If choice == 4:
          Call dequeuerear()
If choice == 5:
          Call display()
If choice == 6:
         Exit
```

10. Stop

Output

```
Queue Operations:
1-->ENQUEUE F
2-->ENQUEUE R
3-->DEQUEUE F
4-->DEQUEUE R
5-->DISPLAY
6-->EXIT

Enter choice: 1
Enter value to enqueue: 10
Queue after ENQUEUE FRONT: 10

Enter choice: 1
Enter value to enqueue: 20
Queue after ENQUEUE FRONT: 20 10

Enter choice: 2
Enter value to enqueue: 30
Queue after ENQUEUE REAR: 20 10 30

Enter choice: 2
Enter value to enqueue: 40
Queue after ENQUEUE REAR: 20 10 30 40

Enter choice: 1
Enter value to enqueue: 50
Queue after ENQUEUE FRONT: 50 20 10 30 40

Enter choice: 2
Enter value to enqueue: 60
Queue Overflow

Enter choice: 5
Queue is: 50 20 10 30 40

Enter choice: 3
Value 50 has been dequeued from FRONT
Queue after DEQUEUE FRONT: 20 10 30 40

Enter choice: 4
Value 40 has been dequeued from REAR
Queue after DEQUEUE REAR: 20 10 30

Enter choice: 3
Value 20 has been dequeued from FRONT
Queue after DEQUEUE FRONT: 10 30

Enter choice: 3
Value 10 has been dequeued from FRONT
Queue after DEQUEUE FRONT: 30

Enter choice: 4
Value 30 has been dequeued from REAR
Queue after DEQUEUE REAR: Queue is Empty

Enter choice: 4
Queue Underflow

Enter choice: 6
Exiting...
```

Result:

Program has been executed successfully and obtained the output