Question:
Design and implement a given type of (ordinary queue, circular queue) queue in C (array implementation/ Linked list implementation). And demonstrate its working with suitable
inputs. Display appropriate messages in case of exceptions.

Aim:
To Implement Circular Queue using Arrays

Algorithm:

Enqueue:

- Firstly check if the queue is empty using the front and rear pointers
- If it's empty set both the front and rear pointers to 0
- Then check if the queue is full by checking the next element of rear corresponds to front
- If it's full display a message which says "overflow"
- If none of the above conditions are satisfied

Dequeue:

- Firstly check if the queue is empty using the front and rear pointers
- If it's empty, throw an error saying that the queue is empty and trying to remove an element is the Underflow condition
- Then check if the queue only has 1 element (this can be done by checking if front and rear pointers are equal)
- If it does indeed have only 1 element, then set the front and rear pointers to -1 indicating that the queue is now empty
- Finally if it doesn't satisfy the above conditions then increment front by 1 and then it's modulus with size

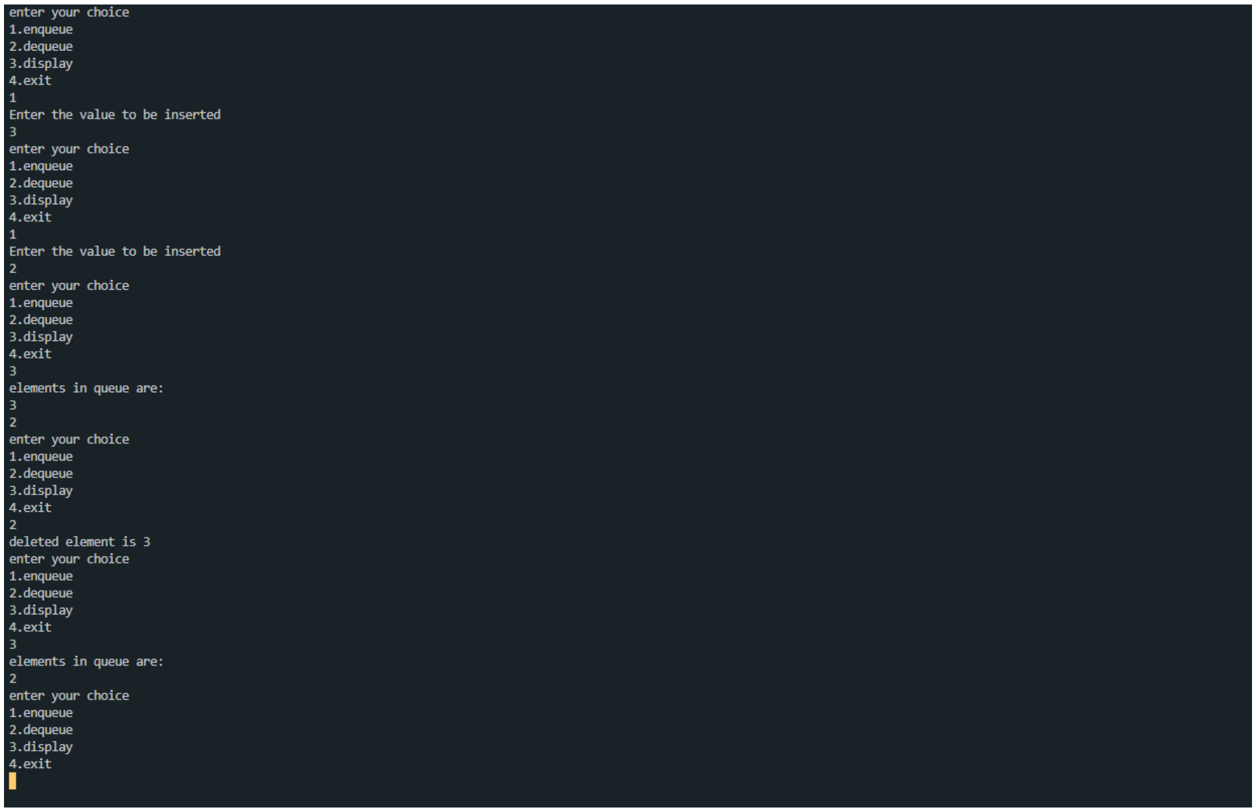
Display

- Firstly check if the queue is empty using the front and rear pointers
- If it is empty, then display a message saying that the queue is empty
- Else display all the elements starting from front to rear using a for loop

Program

```
//implementation of circular queue using array
#include<stdio.h>
#include<stdlib.h>
#define size 5
int queue[size];
int front=-1;
int rear=-1;
void enqueue(int x)  //Inserting value into queue
{
    if(front==-1 && rear==-1)
    {
        front=rear=0;
        queue[rear]=x;
    }
    else if((rear+1)%size==front)
    {
        printf("Queue overflow");
    }
    else
    {
        rear=(rear+1)%size;
        queue[rear]=x;
    }
}

int dequeue() //function to delete element from queue
{
    if(front==-1 && rear==-1)
    {
        printf("Queue overflow");
    }
    else if(front==rear)
    {
        printf("deleted element is%d\n",queue[front]);
        front=rear=-1;
    }
    else
    {
        printf("deleted element is %d\n",queue[front]);
        front=(front+1)%size;
    }
}

void display() //displays the queue
{
    int i=front;
    if(front==-1 && rear==-1)
    {
        printf("Queue is empty");
    }
    else
    {
        printf("elements in queue are:\n");
        while(i<=rear)
        {
            printf("%d\n",queue[i]);
            i=(i+1)%size;
        }
    }
}

int main() //main function
{
    int choice,data;
    while(1)
    {
        printf("enter your choice\n");
        printf("1.enqueue\n2.dequeue\n3.display\n4.exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                    printf("Enter the value to be inserted\n");
                    scanf("%d",&data);
                    enqueue(data);
                    break;

                case 2:
                        dequeue();
                        break;

                case 3:
                    display();
                    break;

                case 4:
                    exit(0);

                default:
                    printf("invalid\n");
        }
    }
}
```

Output



```
enter your choice
1.enqueue
2.dequeue
3.display
4.exit
1
Enter the value to be inserted
3
enter your choice
1.enqueue
2.dequeue
3.display
4.exit
1
Enter the value to be inserted
2
enter your choice
1.enqueue
2.dequeue
3.display
4.exit
3
elements in queue are:
3
2
enter your choice
1.enqueue
2.dequeue
3.display
4.exit
2
deleted element is 3
enter your choice
1.enqueue
2.dequeue
3.display
4.exit
3
elements in queue are:
2
enter your choice
1.enqueue
2.dequeue
3.display
4.exit
```