FAIZAN CHOUDHARY

20BCS021

DSA LAB

16th November 2021

CODE: (code pasted in this format for readability)

```
#include <iostream>
#include <limits.h>
using namespace std;

const int LIMIT=10;
int queue[LIMIT], front=-1, rear=-1;

int isEmpty ()
{
    if (front==-1 && rear==-1)
        return 1;
    else
        return 0;
}

int isFull ()
{
    if ((rear+1) % LIMIT == front)
        return 1;
    else
        return 0;
}

void display ()
{
    if (isEmpty()==1)
        cout<<"\nQueue is empty! Nothing to display\n";
    else
    {
        cout<<"\nQueue elements:\n";
        cout<<" <- ";
        for (int i=0; i<LIMIT; i++)
        {
            if (queue[i]==INT_MAX)
                cout<<"| - |";
            else
                cout<<"| "<<queue[i]<<" |";
        }
        cout<<" -> ";
    }
```

```
    cout<<"\nFront: "<<front<<"\tRear: "<<rear<<endl;
}

void front_rear ()
{
    if (isEmpty()==1)
        cout<<"\nQueue is empty..."<<endl;
    else
    {
        cout<<"\nFront element is: "<<queue[front];
        cout<<"\nRear element is: "<<queue[rear]<<endl;
    }

}

int size ()
{
    if (isEmpty()==1)
        return 0;
    else
        return (rear>front) ? (rear-front+1) : (front-rear+1);
}

void enqueue (int n)
{
    if (isFull()==1)
    {
        cout<<"\nQueue Overflow! Maximum limit (10) reached..."<<endl;
        return;
    }
    if (isEmpty()==1)
    {
        front=rear=0;
        queue[front]=n;
        display();
    }
    else
    {
        rear = (rear+1) % LIMIT;
        queue[rear]=n;
        display();
    }
}

void dequeue ()
{
    if (isEmpty()==1)
    {
        cout<<"\nQueue Underflow! Queue is empty..."<<endl;
        return;
    }
    if (front == rear)
    {
        cout<<"\nDequeuing front element: "<<queue[front]<<endl;
```

```
        queue[front] = INT_MAX;
        front=rear=-1;
        display();
    }
    else
    {
        cout<<"\nDequeuing front element: "<<queue[front]<<endl;
        queue[front] = INT_MAX;
        front++;
        display();
    }
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";

    int ch,n;

    // initialising with default values
    for (int i=0; i<LIMIT; i++)
        queue[i] = INT_MAX;

    while (true)
    {
        A:
        cout<<"\nMENU:\n1. Enqueue\n2. Dequeue\n3. Display front and rear elements\n4.
Check if queue is full\n5. Check if queue is empty\n6. Size of the queue\n7. Display
queue\n8. Exit\n";
        cin>>ch;
        switch (ch)
        {
        case 1: cout<<"\nEnter the element to be enqueued: ";
                cin>>n;
                enqueue(n);
                break;
        case 2: dequeue();
                break;
        case 3: front_rear();
                break;
        case 4: if (isFull()==1)
                    cout<<"\nQueue is full!\n";
                else
                    cout<<"\nQueue is not full.\n";
                break;
        case 5: if (isEmpty()==1)
                    cout<<"\nQueue is empty!\n";
                else
                    cout<<"\nQueue is not empty.\n";
                break;
        case 6: cout<<"\nSize of the queue is: "<<size()<<endl;
                break;
        case 7: cout<<"\nQueue elements: "<<endl;
                display();
```

```
                break;
        case 8: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
        }
    }
    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 22

Queue elements:
 <- | 22 || - || - || - || - || - || - || - || - | ->
Front: 0        Rear: 0
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 33

Queue elements:
 <- | 22 || 33 || - || - || - || - || - || - || - | ->
Front: 0        Rear: 1
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
1

Enter the element to be enqueued: 44

Queue elements:
 <- | 22 || 33 || 44 || - || - || - || - || - || - | ->
Front: 0        Rear: 2
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
3

Front element is: 22
Rear element is: 44
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
6

Size of the queue is: 3
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
4

Queue is not full.
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 22

Queue elements:
 <- | - || 33 || 44 || - || - || - || - || - || - | ->
Front: 1        Rear: 2
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 33

Queue elements:
 <- | - || - || 44 || - || - || - || - || - || - | ->
Front: 2        Rear: 2
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
2

Dequeuing front element: 44

Queue is empty! Nothing to display

Front: -1       Rear: -1
```

```
MENU:
1. Enqueue
2. Dequeue
3. Display front and rear elements
4. Check if queue is full
5. Check if queue is empty
6. Size of the queue
7. Display queue
8. Exit
5

Queue is empty!
```