Data Structure Lab

CEN-391

Program 8

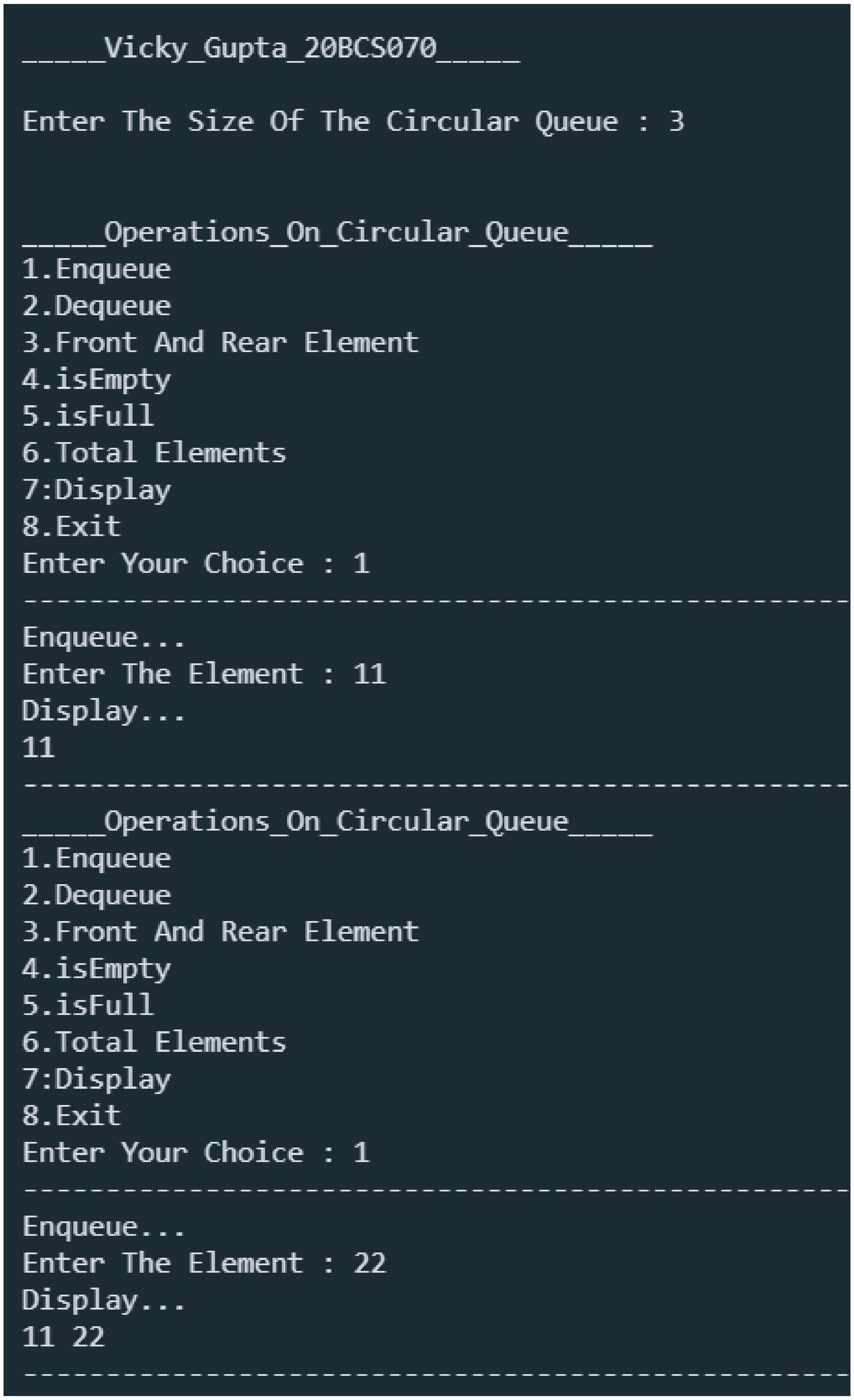
Code :-

```
#include <iostream>
using namespace std;

void isEmpty(int front, int rear)
{
    cout << "isEmpty...\n";
    if (front == -1 && rear == -1)
        cout << "Empty" << endl;
    else
        cout << "Not Empty" << endl;
}

void isFull(int front, int rear, int capacity)
{
    cout << "isFull...\n";
    if ((rear + 1) % capacity == front)
```

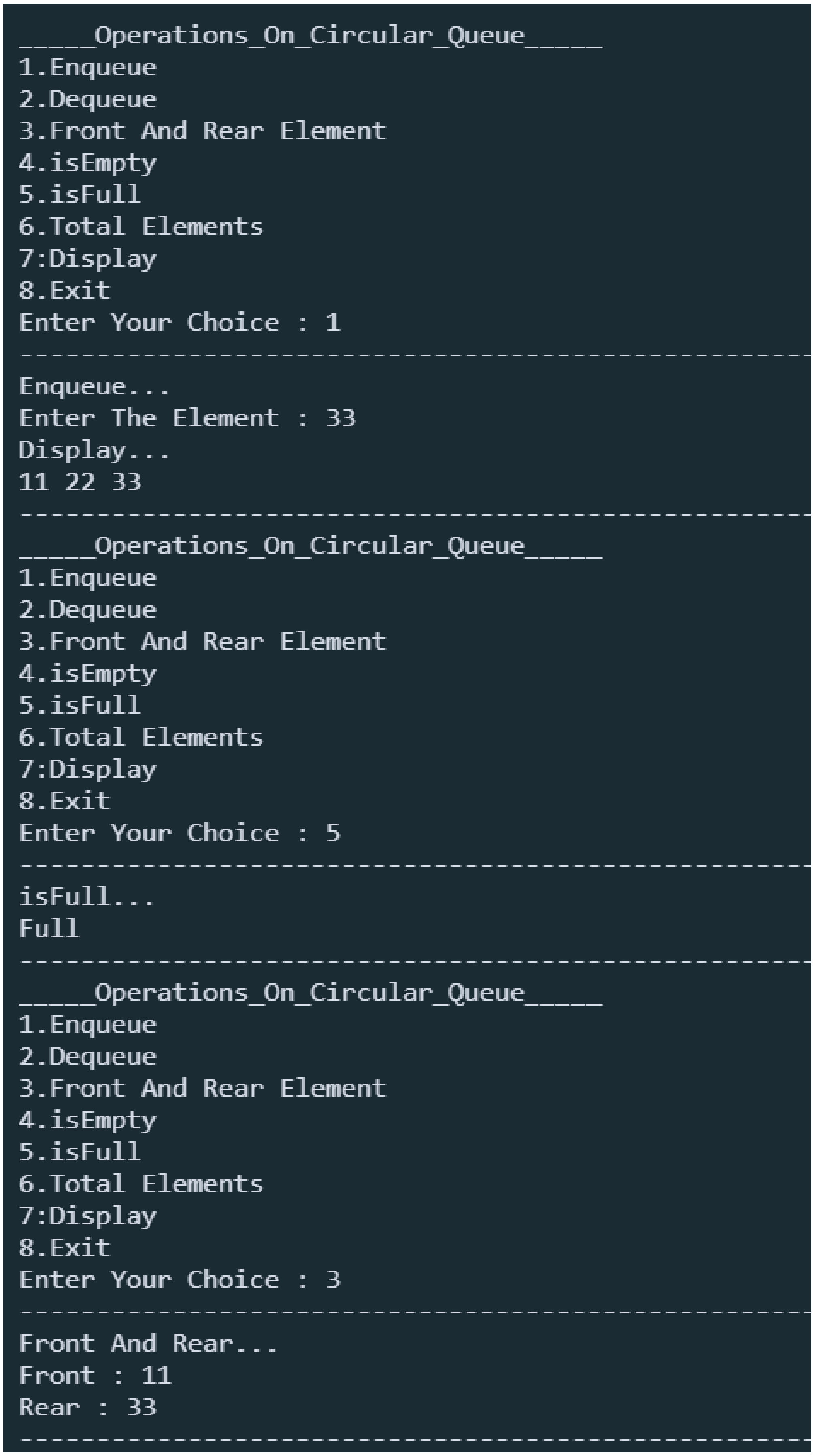
```
        cout << "Full" << endl;
    else
        cout << "Not Full" << endl;
}

void Display(int queue[], int front, int rear, int capacity)
{
    cout << "Display...\n";
    if (rear == -1 && front == -1)
    {
        cout << "Queue Empty" << endl;
        return;
    }
    if (front <= rear)
    {
        for (int i = front; i <= rear; i++)
        {
            cout << queue[i] << " ";
        }
    }
    else
    {
        for (int i = front; i < capacity; i++)
        {
            cout << queue[i] << " ";
        }
        for (int i = 0; i <= rear; i++)
        {
            cout << queue[i] << " ";
        }
    }

    cout << endl;
}

void Enqueue(int queue[], int &front, int &rear, int
capacity)
{
    cout << "Enqueue...\n";
    if (front == -1 && rear == -1)
    {
        front = 0;
        rear = 0;
        cout << "Enter The Element : ";
        cin >> queue[rear];
        Display(queue, front, rear, capacity);
    }
    else if ((rear + 1) % capacity == front)
    {
        cout << "Queue Overflow" << endl;
    }
    else
    {
        rear = (rear + 1) % capacity;
        cout << "Enter The Element : ";
        cin >> queue[rear];
        Display(queue, front, rear, capacity);
    }
}

void Dequeue(int queue[], int &front, int &rear, int
capacity)
{
    cout << "Dequeue...\n";
    if (rear == -1 && front == -1)
    {
        cout << "Queue Underflow" << endl;
    }
    else if (front == rear)
    {
        cout << queue[front] << endl;
        front = -1;
        rear = -1;
        Display(queue, front, rear, capacity);
    }
    else
    {
        cout << queue[front] << endl;
        front = (front + 1) % capacity;
        Display(queue, front, rear, capacity);
    }
}
void Front_Rear(int queue[], int front, int rear)
{
    cout << "Front And Rear...\n";
    if (front == -1 && rear == -1)
    {
        cout << "Queue Is Empty" << endl;
    }
    cout << "Front : " << queue[front] << endl;
    cout << "Rear : " << queue[rear] << endl;
}

void Total_Element(int front, int rear, int capacity)
{
    if (front == -1 && rear == -1)
        cout << "Total Elements In Queue : " << 0 << endl;
    else if (front <= rear)
        cout << "Total Elements In Queue : " << rear - front
+ 1 << endl;
    else
        cout << "Total Elements In Queue : " << front -
capacity + rear + 1 << endl;
}

void Bars()
{
    cout << "-------------------------------------------------
----------------\n";
}
bool Options(int queue[], int &front, int &rear, int
capacity)
{
    int opt;
    cin >> opt;
    Bars();
    switch (opt)
    {
    case 1:
        Enqueue(queue, front, rear, capacity);
        break;
    case 2:
        Dequeue(queue, front, rear, capacity);
        break;
    case 3:
        Front_Rear(queue, front, rear);
        break;
    case 4:
        isEmpty(front, rear);
        break;
    case 5:
        isFull(front, rear, capacity);
        break;
    case 6:
        Total_Element(front, rear, capacity);
        break;
    case 7:
        Display(queue, front, rear, capacity);
        break;
    case 8:
        cout << "Exit...\n";
        return 0;
    default:
        cout << "Invalid Input!\nTry Again!\n";
    }
    Bars();
    return 1;
}

void Menu()
{
    cout << "_____Operations_On_Circular_Queue_____ \n";
    cout << "1.Enqueue \n";
    cout << "2.Dequeue \n";
    cout << "3.Front And Rear Element \n";
    cout << "4.isEmpty \n";
```

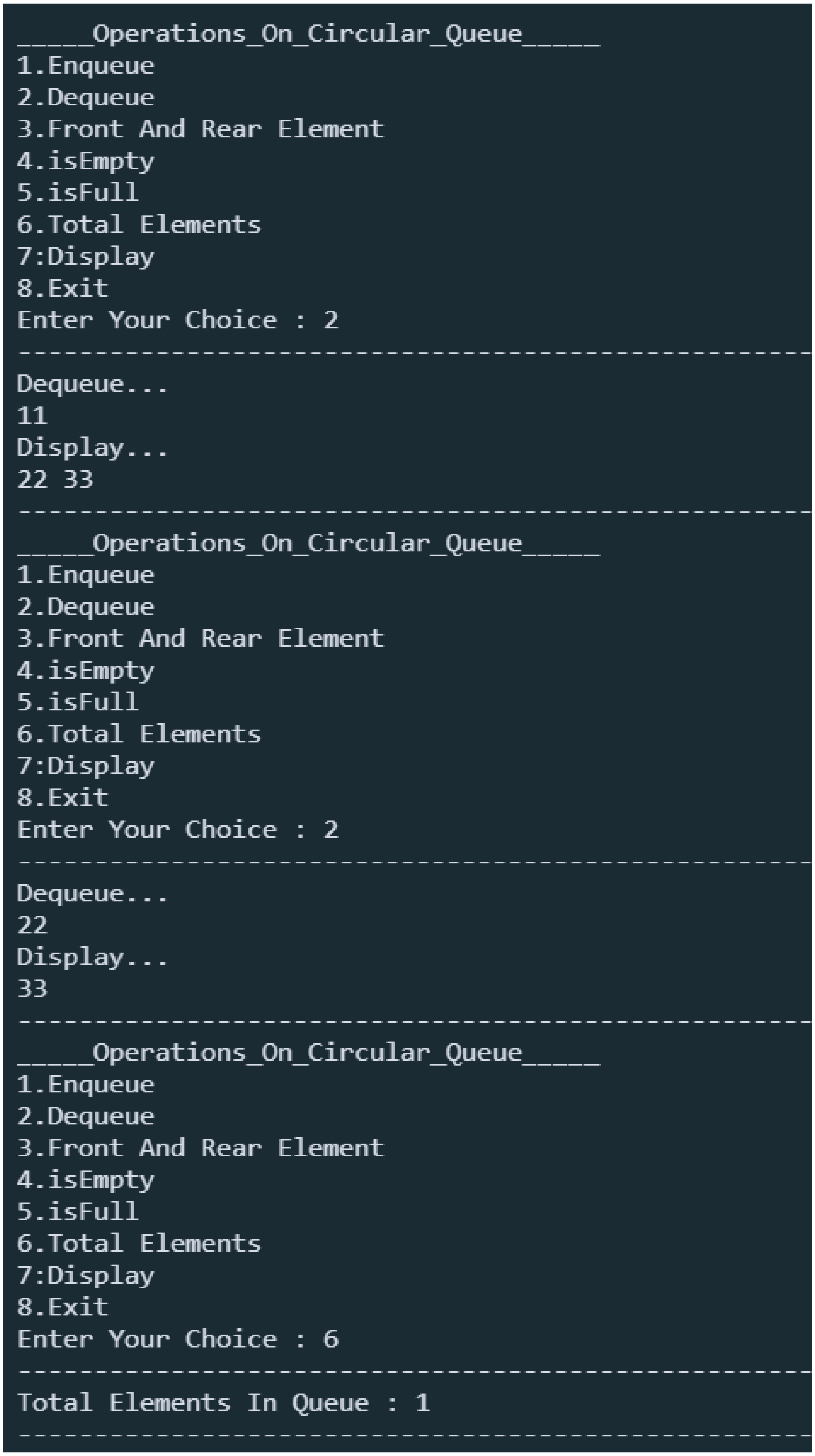
```cpp
    cout << "5.isFull \n";
    cout << "6.Total Elements \n";
    cout << "7:Display \n";
    cout << "8.Exit \n";
    cout << "Enter Your Choice : ";
}

int main()
{

    system("cls");
    cout << "_____Vicky_Gupta_20BCS070_____\n\n";
    cout << "Enter The Size Of The Circular Queue : ";
    int capacity, front = -1, rear = -1;
    cin >> capacity;
    int *queue = (int *)malloc(sizeof(int) * capacity);
    cout << "\n\n";
    while (true)
    {
        Menu();
        if (!Options(queue, front, rear, capacity))
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

Output :-

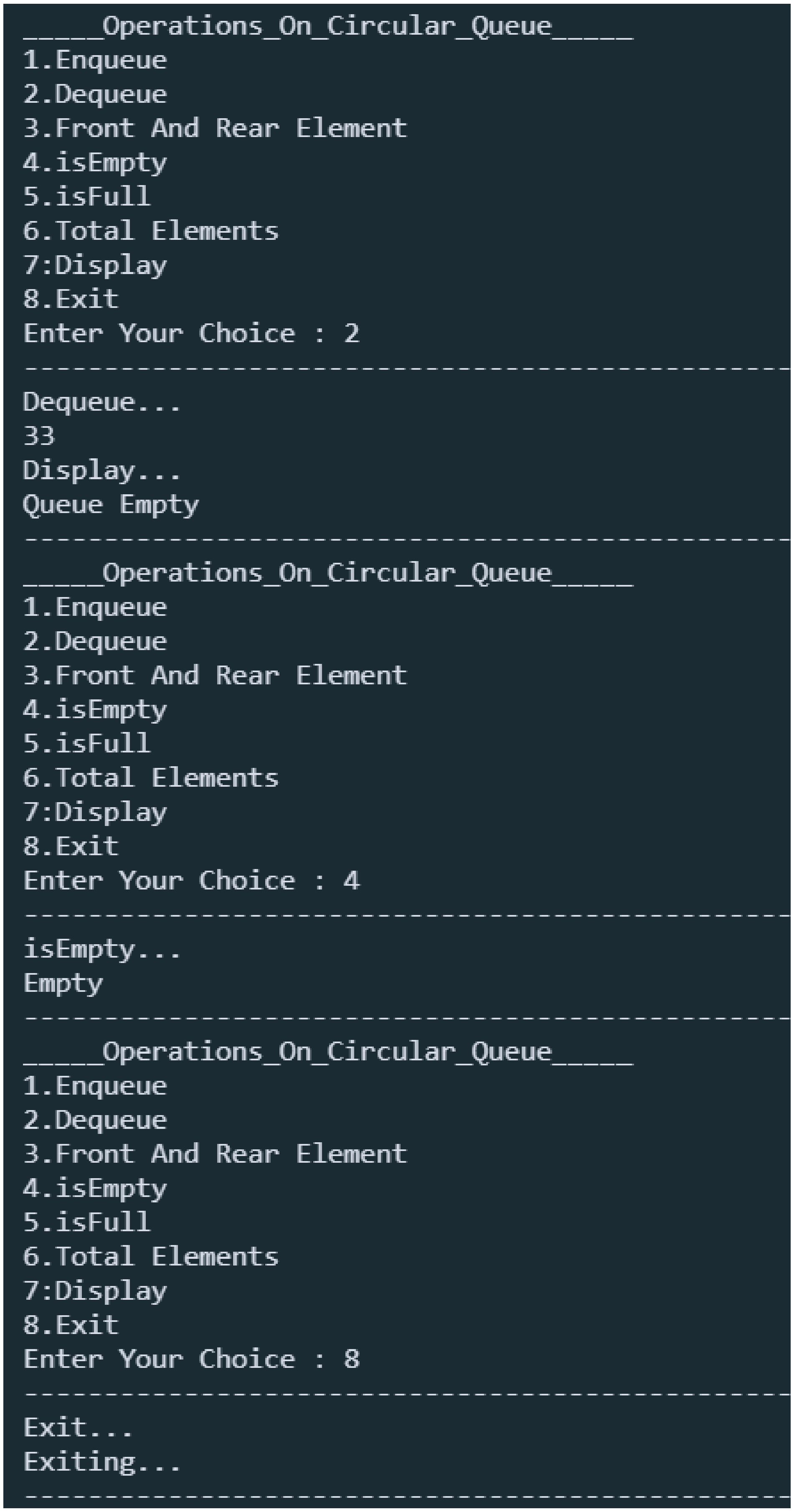


```
_____Vicky_Gupta_20BCS070_____

Enter The Size Of The Circular Queue : 3


_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
--------------------------------------------------
Enqueue...
Enter The Element : 11
Display...
11
--------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
--------------------------------------------------
Enqueue...
Enter The Element : 22
Display...
11 22
--------------------------------------------------
```


```
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
-----------------------------------------------------
Enqueue...
Enter The Element : 33
Display...
11 22 33
-----------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 5
-----------------------------------------------------
isFull...
Full
-----------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 3
-----------------------------------------------------
Front And Rear...
Front : 11
Rear : 33
-----------------------------------------------------
```



```
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
----------------------------------------------------
Dequeue...
11
Display...
22 33
----------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
----------------------------------------------------
Dequeue...
22
Display...
33
----------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 6
----------------------------------------------------
Total Elements In Queue : 1
----------------------------------------------------
```


```
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
-----------------------------------------------
Dequeue...
33
Display...
Queue Empty
-----------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 4
-----------------------------------------------
isEmpty...
Empty
-----------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 8
-----------------------------------------------
Exit...
Exiting...
-----------------------------------------------
```