Name – Yash Daga Roll Number – 20BCE7323

1) Write a program to implement following modified circular queue.

a) Enqueue()--> insert an element at rear side

b) Dequeue()--> remove a maximum element from queue

Ans: Code:

```
class CircularQueue{
private int size, front, rear;
private int[] queue;
CircularQueue(int size)
{
	this.size = size;
	this.front = this.rear = -1;
	queue = new int[size];
}
public void Enqueue(int data)
{
	if((front == 0 && rear == size - 1) ||
	(rear == (front - 1) % (size - 1)))
	{
		System.out.print("Queue is Full");
	}
	else if(front == -1)
	{
		front = 0;
		rear = 0;
		queue[rear] = data;
```

```
        }
        else if(rear == size - 1 && front != 0)
        {
                rear = 0;
                queue[rear] = data;
        }
        else
        {
                rear = (rear + 1);
                if(front <= rear)
                {
                        queue[rear] = data;
                        //queue.add(rear, data);
                }
                else
                {
                        queue[rear] = data;
                }
        }
}
public int Dequeue()
{
        int temp;
        if(front == -1)
        {
                System.out.print("Queue is Empty");
```

```
            return -1;
        }

        //temp = queue.get(front);
        temp = queue[front];
        if(front == rear)
        {
            front = -1;
            rear = -1;
        }

        else if(front == size - 1)
        {
            front = 0;
        }
        else
        {
            front = front + 1;
        }
        return temp;
    }
    public void displayQueue()
    {
        if(front == -1)
        {
            System.out.print("Queue is Empty");
```

```
            return;
        }
        System.out.print("Elements in the " +
                                    "circular queue are: ");

        if(rear >= front)
        {
            for(int i = front; i <= rear; i++)
            {
                System.out.print(/*queue.get(i)*/queue[i]);
                System.out.print(" ");
            }
            System.out.println();
        }
        else
        {
            for(int i = front; i < size; i++)
            {
                System.out.print(/*queue.get(i)*/queue[i]);
                System.out.print(" ");
            }
            for(int i = 0; i <= rear; i++)
            {
                System.out.print(/*queue.get(i)*/queue[i]);
                System.out.print(" ");
            }
```

```
                System.out.println();
        }
}
public static void main(String[] args)
{
        CircularQueue q = new CircularQueue(5);

        q.Enqueue(7);
        q.Enqueue(11);
        q.Enqueue(6);
        q.Enqueue(3);

        q.displayQueue();

        int x = q.Dequeue();
        if(x != -1)
        {
                System.out.print("Deleted value = ");
                System.out.println(x);
        }

        x = q.Dequeue();
        if(x != -1)
        {
                System.out.print("Deleted value = ");
                System.out.println(x);
```

```
        }

        q.displayQueue();

        q.Enqueue(9);
        q.Enqueue(20);
        q.Enqueue(5);

        q.displayQueue();

        q.Enqueue(20);
    }
}
```

Command Prompt

```
D:\20BCE7323>javac CircularQueue.java

D:\20BCE7323>java CircularQueue
Elements in the circular queue are: 7 11 6 3
Deleted value = 7
Deleted value = 11
Elements in the circular queue are: 6 3
Elements in the circular queue are: 6 3 9 20 5
Queue is Full
D:\20BCE7323>
```

2) Write a program to read a number (n) and extract digit by digit from n and enqueue() into queue. By using dequeue() find reverse of n.

Ans: Code:

```
import java.util.*;

class ReverseNumber{

private int size, front, rear;

private int[] queue;

ReverseNumber(int size)

{

        this.size = size;

        this.front = this.rear = -1;

        queue = new int[size];

}

public void Enqueue(int data)

{

        if((front == 0 && rear == size - 1) ||

        (rear == (front - 1) % (size - 1)))

        {

                System.out.print("Queue is Full");

        }

        else if(front == -1)

        {

                front = 0;

                rear = 0;

                //queue.add(rear, data);

                queue[rear] = data;

        }
```

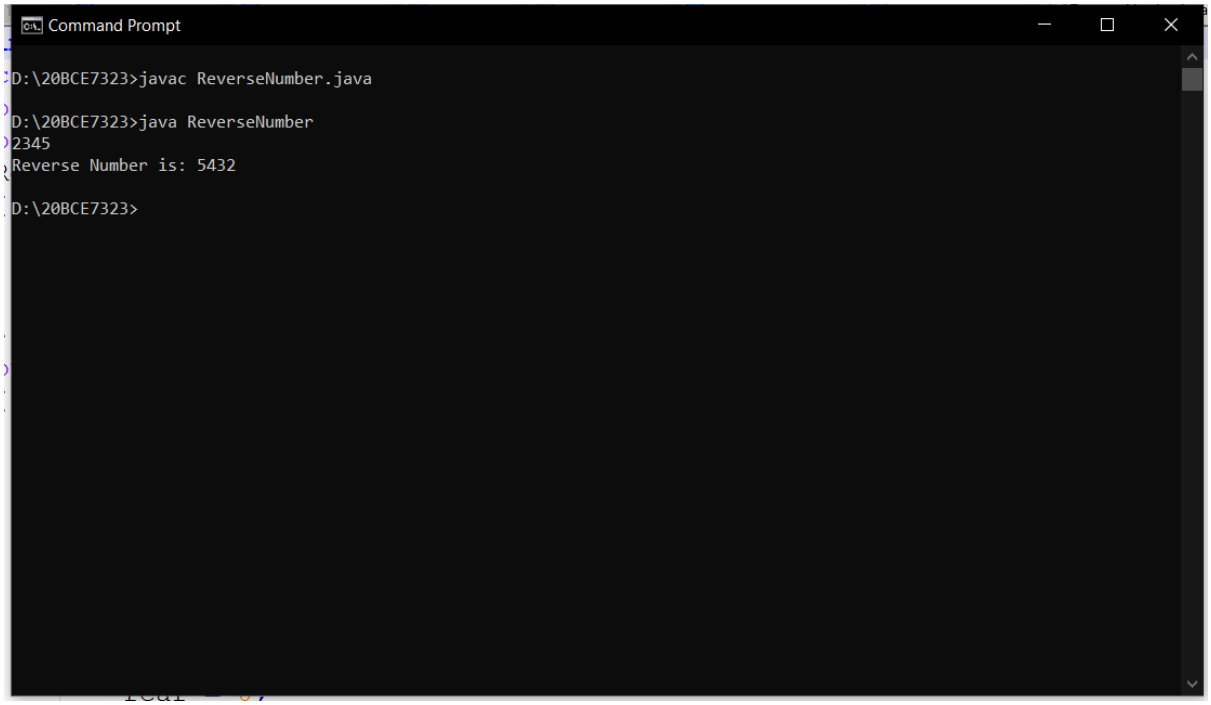
```
        else if(rear == size - 1 && front != 0)
        {
                rear = 0;
                //queue.set(rear, data);
                queue[rear] = data;
        }
        else
        {
                rear = (rear + 1);
                if(front <= rear)
                {
                        queue[rear] = data;
                        //queue.add(rear, data);
                }
                else
                {
                        queue[rear] = data;
                        //queue.set(rear, data);
                }
        }
}
public int Dequeue()
{
        int temp;
        if(front == -1)
        {
```

```
            System.out.print("Queue is Empty");
            return -1;
        }

        //temp = queue.get(front);
        temp = queue[front];
        if(front == rear)
        {
            front = -1;
            rear = -1;
        }

        else if(front == size - 1)
        {
            front = 0;
        }
        else
        {
            front = front + 1;
        }
        return temp;
    }
    public void displayQueue()
    {
        if(front == -1)
        {
```

```
        System.out.print("Queue is Empty");
        return;
    }
    System.out.print("Elements in the " +
                                "circular queue are: ");

    if(rear >= front)
    {
        for(int i = front; i <= rear; i++)
        {
            System.out.print(/*queue.get(i)*/queue[i]);
            System.out.print(" ");
        }
        System.out.println();
    }
    else
    {
        for(int i = front; i < size; i++)
        {
            System.out.print(/*queue.get(i)*/queue[i]);
            System.out.print(" ");
        }
        for(int i = 0; i <= rear; i++)
        {
            System.out.print(/*queue.get(i)*/queue[i]);
            System.out.print(" ");
```

```
                }
                System.out.println();
        }
}
boolean isEmpty() {
   if (front == -1)
     return true;
   else
     return false;
}
public static void main(String[] args)
{
        ReverseNumber q = new ReverseNumber(5);
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int temp = n;
        int digit;
        while(temp > 0) {
                digit = temp % 10;
                q.Enqueue(digit);
                temp /= 10;
        }
        temp = 0;
        while(!q.isEmpty()) {
                digit = q.Dequeue();
                temp = temp * 10 + digit;
```

```
        }
        System.out.println("Reverse Number is: " + temp);
    }
}
```


Command Prompt
D:\20BCE7323>javac ReverseNumber.java
D:\20BCE7323>java ReverseNumber
2345
Reverse Number is: 5432
D:\20BCE7323>

2) Write a program to read a number (n) and extract digit by digit from n and enqueue() into queue. By using dequeue() find reverse of n.

Ans Code: