NAME:SHEETAL

ROLL NO: 11172531

**DATA\_STRUCTURE\_ASSESSMENT:-**

**SECTION-A:**

**Answers:**

1. **Stack**
2. **I ,II,III**
3. **O(Log n)**
4. **())(()**
5. **6,1**
6. **None of the above**
7. **142**
8. **146641**
9. **(c)delete the last node of the linked list (d)add an element at the end of the linked list.**
10. **All the above**

**SECTION-B(CODING)**

|  |
| --- |
| class LinkedList  {  static class Node  {      int data;      Node next;  };    static Node circular(Node head)  {      Node start = head;        while (head.next != null)          head = head.next;          head.next = start;       return start;  }    static Node push(Node head, int data)  {      Node newNode = new Node();            newNode.data = data;    newNode.next = (head);     (head) = newNode;        return head;  }    static void displayList( Node node)  {      Node start = node;        while (node.next != start)      {          System.out.print(" "+ node.data);          node = node.next;      }          System.out.print(" " + node.data);  }    public static void main(String args[])  {        Node head = null;        head = push(head, 15);      head = push(head, 14);      head = push(head, 13);      head = push(head, 22);      head = push(head, 17);        circular(head);        System.out.print("Display list: \n");      displayList(head);  }  }    12) |

class Queue {

    private static int front, rear, capacity;

    private static int queue[];

    Queue(int c)

    {

        front = rear = 0;

        capacity = c;

        queue = new int[capacity];

    }

    static void queueEnqueue(int data)

    {

        if (capacity == rear) {

            System.out.printf("\nQueue is full\n");

            return;

        }

        else {

            queue[rear] = data;

            rear++;

        }

        return;

    }

    static void queueDequeue()

    {

        if (front == rear) {

            System.out.printf("\nQueue is empty\n");

            return;

        }

        else {

            for (int i = 0; i < rear - 1; i++) {

                queue[i] = queue[i + 1];

            }

            if (rear < capacity)

                queue[rear] = 0;

            // decrement rear

            rear--;

        }

        return;

    }

        static void queueDisplay()

    {

        int i;

        if (front == rear) {

            System.out.printf("\nQueue is Empty\n");

            return;

        }

            System.out.printf(" %d <-- ", queue[i]);

        }

        return;

    }

}

public class StaticQueueinjava {

    public static void main(String[] args)

    {

        Queue q = new Queue(4);

        q.queueDisplay();

        q.queueEnqueue(20);

        q.queueEnqueue(30);

        q.queueEnqueue(40);

        q.queueEnqueue(50)

 q.queueEnqueue(60);

        q.queueDisplay();

        q.queueDequeue();

        q.queueDequeue();

        System.out.printf("\n\nafter two node deletion\n\n");

         q.queueDisplay();

 q.queueFront();

    }

}