

LAB EXAM

DATA STRUCTURE AND ALGORITHMS

Write a Java program to

a. Perform insertion sort

```
package DataStructureLabExam;
import java.util.Scanner;
public class InserationSort {

    public static void main(String[] args)
    {
        int n, i, j, element;
        Scanner sn = new Scanner(System.in);

        System.out.print("Enter the Size of Array: ");
        n = sn.nextInt();
        int[] arr = new int[n];
        System.out.print("Enter " +n+ " Elements: ");
        for(i=0; i<n; i++)
            arr[i] = sn.nextInt();

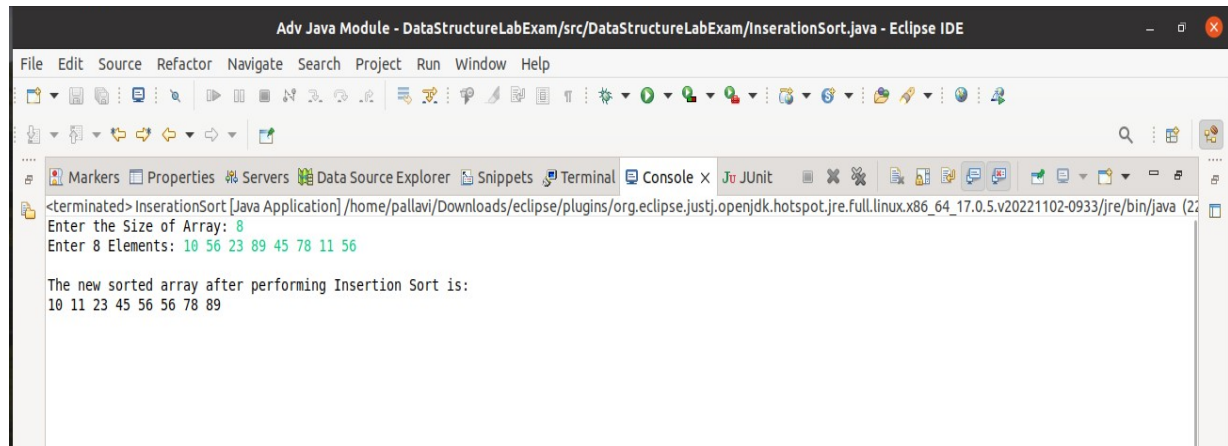
        for(i=1; i<n; i++)
        {
            element = arr[i];

            for(j=(i-1); j>=0 && arr[j]>element; j--)
                arr[j+1] = arr[j];

            arr[j+1] = element;
        }

        System.out.println("\nThe new sorted array after performing Insertion Sort
is: ");
        for(i=0; i<n; i++)
            System.out.print(arr[i]+ " ");
    }
}
```

Output:



The screenshot shows the Eclipse IDE interface with a Java application named 'InsertionSort'. The console output is as follows:

```
<terminated> InsertionSort [Java Application] /home/pallavi/Downloads/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.5.v20221102-0933/jre/bin/java (2)
Enter the Size of Array: 8
Enter 8 Elements: 10 56 23 89 45 78 11 56

The new sorted array after performing Insertion Sort is:
10 11 23 45 56 56 78 89
```

b. Implement queue using array

```
package DataStructureLabExam;
import java.util.*;
```

```
public class QueueArray {
    static private int front, rear, capacity;
    static private int queue[];
    public QueueArray(int c) {
        front = 0;
        rear = 0;
        capacity = c;
        queue = new int[capacity];
    }
    // at the rear of the queue
    static void queueEnqueue(int data) {
        if (capacity == rear) {
            System.out.printf("\nQueue is full\n");
            return;
        }
        else {
            queue[rear] = data;
            rear++;
        }
        return;
    }
    // function to delete an element
```

```

static void queueDequeue() {
    if (front == rear) {
        System.out.printf("Queue is empty\n");
        return;
    }
    else {
        for (int i = 0; i < rear - 1; i++) {
            queue[i] = queue[i + 1];
        }
        rear--;
    }
    return;
}
// print queue elements
static void queueDisplay()
{
    int i;
    if (front == rear) {
        System.out.printf("Queue is Empty\n");
        return;
    }
    for (i = front; i < rear; i++) {
        System.out.printf(" %d ", queue[i]);
    }
    return;
}
}

```

```
package DataStructureLabExam;
```

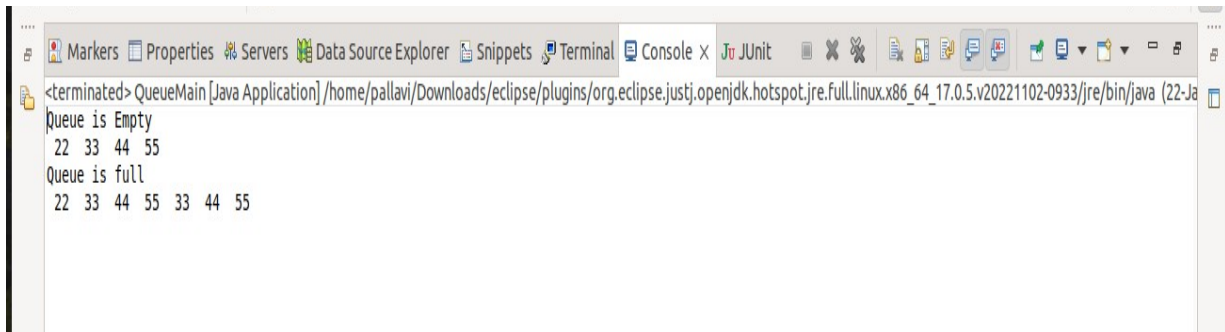
```

public class QueueMain {
    public static void main(String[] args) {
        QueueArray q = new QueueArray(4);
        q.queueDisplay();
        // inserting elements in the queue
        q.queueEnqueue(22);
        q.queueEnqueue(33);
        q.queueEnqueue(44);
        q.queueEnqueue(55);
        // print Queue elements
        q.queueDisplay();
        // insert element in the queue
        q.queueEnqueue(66);
    }
}

```

```
        // print Queue elements
        q.queueDisplay();
        q.queueDequeue();
        q.queueDisplay();
    }
}
```

Output:



```
<terminated> QueueMain [Java Application] /home/pallavi/Downloads/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.5.v20221102-0933/jre/bin/java (22-Ja
Queue is Empty
22 33 44 55
Queue is full
22 33 44 55 33 44 55
```