Tejas Khadke

DSA Lab Exam

1)

```
package com.test;

import java.util.Scanner;

class AdjMatrixWeightedGraph {
    public static final int INF = 999;
    private int vertCount;
    private int edgeCount;
    private int [][] adjmat;

    public AdjMatrixWeightedGraph(int
vertexCount) {
        edgeCount = 0;
        vertCount = vertexCount;
        adjmat = new
int[vertCount][vertCount];
        for (int i = 0; i < vertCount; i++) {
            for (int j = 0; j < vertCount;
j++)
                adjmat[i][j] = INF;
        }
    }

    public void accept(Scanner sc) {
        System.out.print("Enter number of
edges: ");
        edgeCount = sc.nextInt();
        for (int i = 0; i < edgeCount; i++) {
```

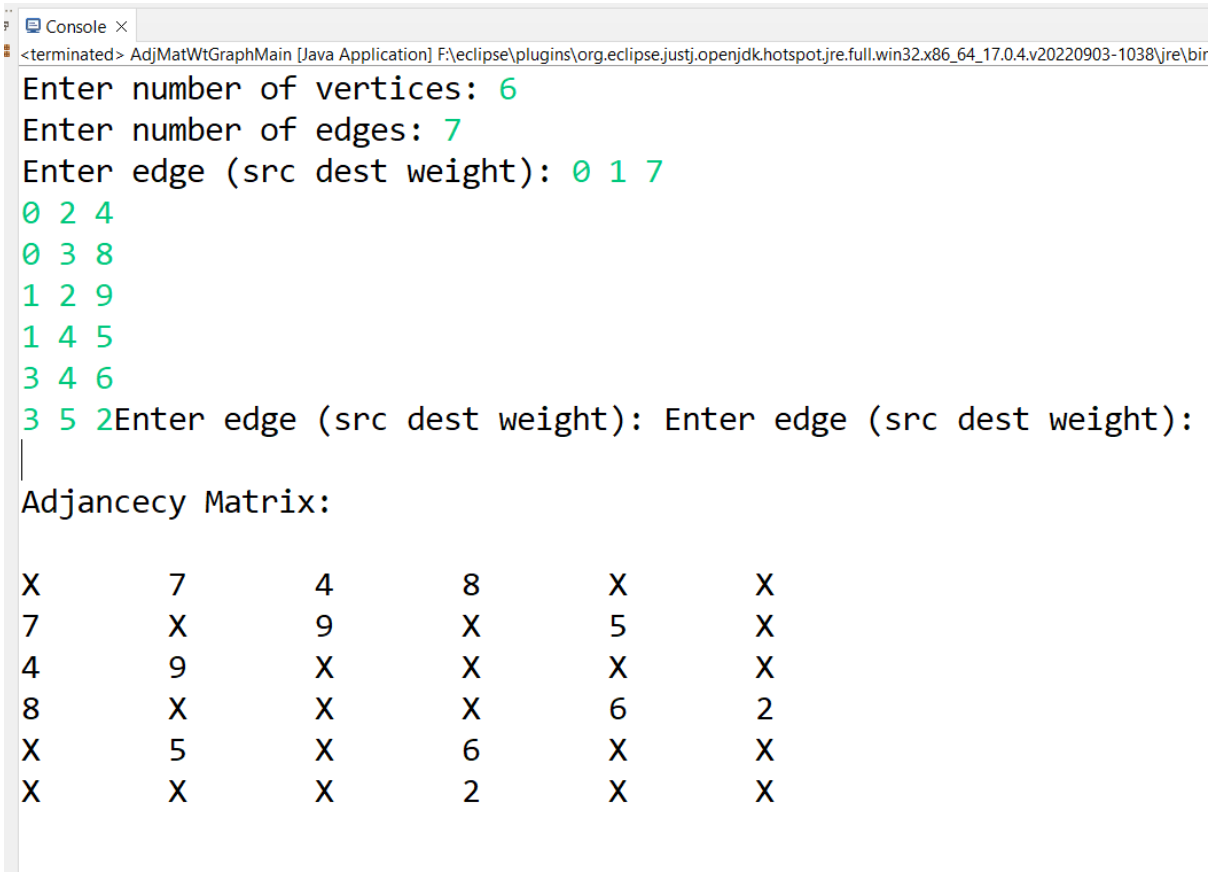
```
            System.out.print("Enter edge (src
dest weight): ");
            int src = sc.nextInt();
            int dest = sc.nextInt();
            int wt = sc.nextInt();
            adjmat[src][dest] = wt;
            adjmat[dest][src] = wt; // delete
this line for directed graph.
        }
    }

    public void display() {
        System.out.println("\nAdjancecy
Matrix: \n");
        for (int i = 0; i < vertCount; i++) {
            for (int j = 0; j < vertCount;
j++) {
                if(adjmat[i][j] == INF)
                    System.out.print("X\t");
                else

    System.out.print(adjmat[i][j] + "\t");
            }
            System.out.println();
        }
    }
}

public class AdjMatWtGraphMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of
vertices: ");
```

```
        int vertCount = sc.nextInt();
        AdjMatrixWeightedGraph g = new
AdjMatrixWeightedGraph(vertCount);
        g.accept(sc);
        g.display();
        sc.close();
    }
}
```



2)

```
package com.test;
```

```
import static java.lang.System.exit;
public class Stack {

        public static void main(String[] args)
        {
            StackUsingLinkedlist obj
                = new StackUsingLinkedlist();
            obj.push(11);
            obj.push(22);
            obj.push(33);
            obj.push(44);


            obj.display();


            System.out.printf("\nTop element
is %d\n",
                              obj.peek());


            obj.pop();
            obj.pop();


            obj.display();

            System.out.printf("\nTop element
is %d\n",obj.peek());

        }
    }
```

```
    class StackUsingLinkedlist {

        private class Node {

            int data;
            Node link;
        }

        Node top;

        StackUsingLinkedlist() { this.top =
null; }


        public void push(int x)
        {

            Node temp = new Node();


            if (temp == null) {
                System.out.print("\nHeap
Overflow");
                return;
            }


            temp.data = x;
            temp.link = top;


            top = temp;
        }
```

```
        public boolean isEmpty() { return top
== null; }

        public int peek()
        {

            if (!isEmpty()) {
                return top.data;
            }
            else {
                System.out.println("Stack is
empty");
                return -1;
            }
        }


        public void pop()
        {

            if (top == null) {
                System.out.print("\nStack
Underflow");
                return;
            }

            top = (top).link;
        }

        public void display()
        {
```

```
            if (top == null) {
                System.out.printf("\nStack
Underflow");
                exit(1);
            }
            else {
                Node temp = top;
                while (temp != null) {



    System.out.print(temp.data);



                    temp = temp.link;
                    if(temp != null)
                        System.out.print(" ->
");
                }
            }
        }
    }
```