Name: Vijay Misal

Div: C Batch: C3

Roll No: 233073

PRN No: 22320079

**Practical No: 6**

**Title:** A business house has several offices in different countries; they want to lease phone lines to connect them with each other and the phone company charges different rent to connect different pairs of cities. Business house want to connect all its offices with a minimum total cost. Represent using appropriate data structure. Apply Prim’s algorithm to find minimum total cost

**Code:**

import **java**.**util**.**Arrays**;

import **java**.**util**.**Scanner**;

public class **PRactical6** {

    public int[][] adjacencyMatrix;

    public int numVertices;

 public **PRactical6**(int numVertices) {

 this.numVertices = numVertices;

 adjacencyMatrix = new int[numVertices][numVertices];

 }

    public void **addEdge**(int source, int destination, int weight) {

        adjacencyMatrix[source][destination] = weight;

        adjacencyMatrix[destination][source] = weight; *// For undirected graph*

    }

    public void **printGraph**() {

**System**.out.**println**("Graph (Adjacency Matrix):");

        for (int i = 0; i < numVertices; i++) {

            for (int j = 0; j < numVertices; j++) {

**System**.out.**print**(adjacencyMatrix[i][j] + " ");

            }

**System**.out.**println**();

        }

    }

    public void **prims**(int source) {

        int[] parent = new int[numVertices];

        int[] key = new int[numVertices];

        boolean[] mstSet = new boolean[numVertices];

**Arrays**.**fill**(key, **Integer**.MAX\_VALUE);

**Arrays**.**fill**(mstSet, false);

        key[source] = 0;

        parent[source] = -1;

        for (int count = 0; count < numVertices - 1; count++) {

            int u = **minKey**(key, mstSet);

            mstSet[u] = true;

            for (int v = 0; v < numVertices; v++) {

                if (adjacencyMatrix[u][v] != 0 && !mstSet[v] && adjacencyMatrix[u][v] < key[v]) {

                    parent[v] = u;

                    key[v] = adjacencyMatrix[u][v];

                }

            }

        }

**printMST**(parent);

    }

    public int **minKey**(int[] key, boolean[] mstSet) {

        int min = **Integer**.MAX\_VALUE, minIndex = -1;

        for (int v = 0; v < numVertices; v++) {

            if (!mstSet[v] && key[v] < min) {

                min = key[v];

                minIndex = v;

            }

        }

        return minIndex;

    }

    public void **printMST**(int[] parent) {

**System**.out.**println**("Minimum Spanning Tree (Prim's Algorithm):");

        for (int i = 1; i < numVertices; i++) {

**System**.out.**println**("Edge: " + parent[i] + " - " + i + " Weight: " +

                    adjacencyMatrix[i][parent[i]]);

        }

    }

    public static void **main**(**String**[] args) {

**Scanner** scanner = new **Scanner**(**System**.in);

**System**.out.**print**("Enter the number of vertices: ");

        int numVertices = scanner.**nextInt**();

**PRactical6** graph = new **PRactical6**(numVertices);

**System**.out.**println**("Enter edges in the format: source destination weight");

        while (true) {

            int source = scanner.**nextInt**();

            int destination = scanner.**nextInt**();

            int weight = scanner.**nextInt**();

            if (source == -1 || destination == -1 || weight == -1) {

                break;

            }

            if (source < 0 || source >= numVertices || destination < 0 || destination >= numVertices) {

**System**.out.**println**("Invalid vertex index. Vertex index should be between 0 and " +

                        (numVertices - 1));

                continue;

            }

            graph.**addEdge**(source, destination, weight);

        }

        graph.**printGraph**();

**System**.out.**print**("Enter the source vertex for Prim's algorithm: ");

        int sourceVertex = scanner.**nextInt**();

        graph.**prims**(sourceVertex);

        scanner.**close**();

    }

}

**Output:**Enter the number of vertices: 4

Enter edges in the format: source destination weight

0 1 2

0 2 4

1 2 1

2 3 3

-1 -1 -1

Graph (Adjacency Matrix):

0 2 4 0

2 0 1 0

4 1 0 3

0 0 3 0

Enter the source vertex for Prim's algorithm: 0

Minimum Spanning Tree (Prim's Algorithm):

Edge: 0 - 1 Weight: 2

Edge: 1 - 2 Weight: 1

Edge: 2 - 3 Weight: 3