PRACTICAL NO. 2

Roll no.: 14

Batch: B1

Name: Abhishek Angadi

Aim: Construction of Minimum Spanning Tree.

PART-A Problem Statement: Little Richie connects the freckles on his Dad's back to form a picture of the Liberty Bell. Consider Dad's back to be a plane with freckles at various (x, y) locations. Your job is to tell Richie how to connect the dots so as to minimize the amount of ink used. Richie connects the dots by drawing straight lines between pairs, without lifting the pen between lines. When Richie is done there must be a sequence of connected lines from any freckle to any other freckle.

- Consider the distance between freckles as input in the form of matrix from the user.
- Apply minimum spanning tree algorithm and print the names of connected freckles (F1, F2, F3, etc.) along with the distance between them and a total value.

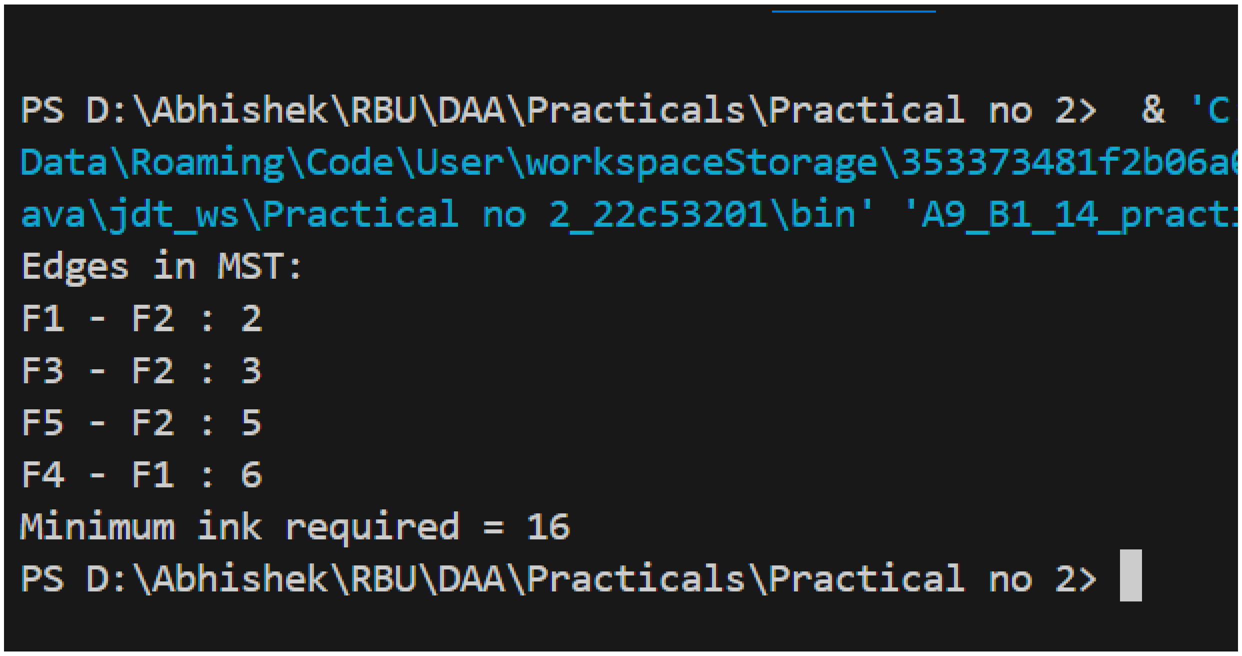
Code:

```
public class A9_B1_14_practical2A {
    static final int INF = Integer.MAX_VALUE;

    public static void primMST(int cost[][], int n) {
        int[] near = new int[n];
        int[][] t = new int[n - 1][2];
        int mincost = 0;

        int k = -1, l = -1, min = INF;
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                if (cost[i][j] < min) {
                    min = cost[i][j];
                    k = i;
                    l = j;
        }
    }
}

mincost = cost[k][l];
t[0][0] = k;
t[0][1] = l;

for (int i = 0; i < n; i++) {
    if (cost[i][k] < cost[i][l])
        near[i] = k;
    else
        near[i] = l;
}

near[k] = near[l] = -1;

for (int i = 1; i < n - 1; i++) {
    int j = -1;
    min = INF;
    for (int v = 0; v < n; v++) {
        if (near[v] != -1 && cost[v][near[v]] < min) {
            min = cost[v][near[v]];
            j = v;
        }
    }

    t[i][0] = j;
    t[i][1] = near[j];
    mincost += cost[j][near[j]];
```

```
            near[j] = -1;

            for (int v = 0; v < n; v++) {
                if (near[v] != -1 && cost[v][j] < cost[v][near[v]]) {
                    near[v] = j;
                }
            }
        }

        System.out.println("Edges in MST:");
        for (int i = 0; i < n - 1; i++) {
            int u = t[i][0];
            int v = t[i][1];
            System.out.println("F" + (u + 1) + " - F" + (v + 1) + " : " + cost[u][v]);
        }
        System.out.println("Minimum ink required = " + mincost);
    }

    public static void main(String[] args) {
        int INF = Integer.MAX_VALUE;
        int[][] cost = {
            {INF, 2, INF, 6, INF},
            {2, INF, 3, 8, 5},
            {INF, 3, INF, INF, 7},
            {6, 8, INF, INF, 9},
            {INF, 5, 7, 9, INF}
        };

        int n = cost.length;
        primMST(cost, n);
```

```
  }
}
```

Output:



```
PS D:\Abhishek\RBU\DAA\Practicals\Practical no 2>  & 'C
Data\Roaming\Code\User\workspaceStorage\353373481f2b06a
ava\jdt_ws\Practical no 2_22c53201\bin' 'A9_B1_14_pract
Edges in MST:
F1 - F2 : 2
F3 - F2 : 3
F5 - F2 : 5
F4 - F1 : 6
Minimum ink required = 16
PS D:\Abhishek\RBU\DAA\Practicals\Practical no 2>
```

PART-B: Problem Statement: A telecommunications organization has offices spanned across multiple locations around the globe. It has to use leased phone lines for connecting all these offices with each other. The organization, wants to use minimum cost for connecting all its offices. This requires that all the offices should be connected using a minimum number of leased lines so as to reduce the effective cost. A.

Consider the following for deciding connections in same state in India:

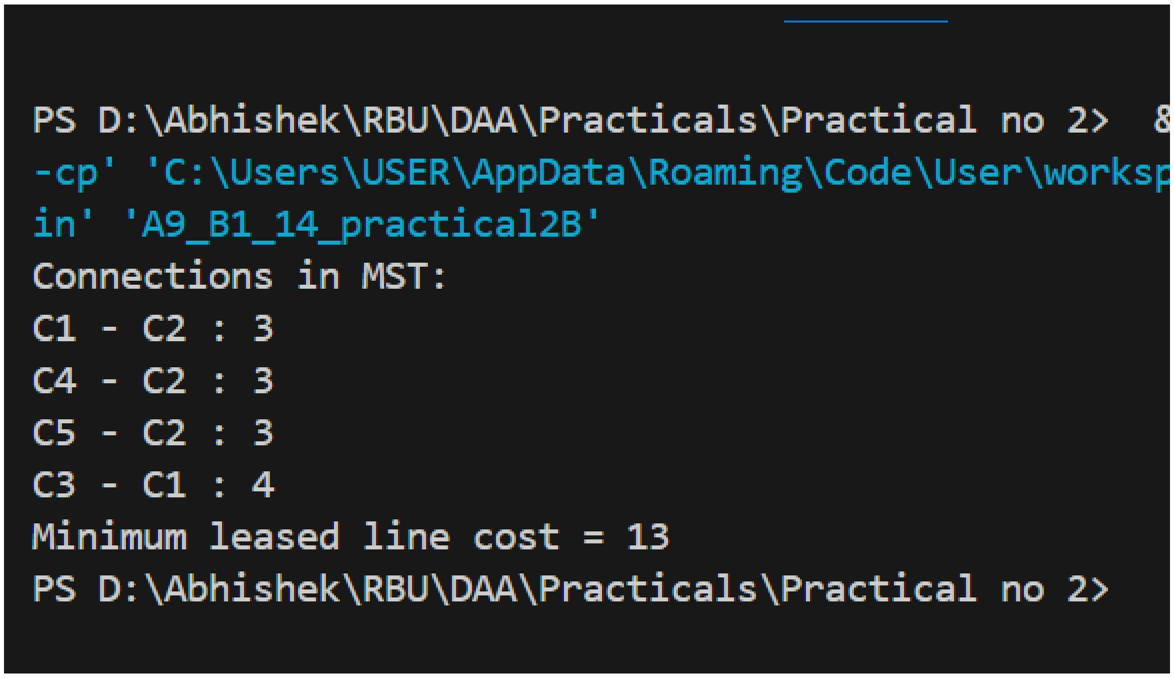
i. Find the latitude and longitude of cities in same state. Consider 4 to 6 cities.
ii. Calculate the cost of connecting each pair of offices by computing the distance between different pair of different cities (as considered in part A) and construct a fully connected graph.
iii. Compute a minimum spanning tree using either Prims or Kruskals Method to find the cost of connecting offices in different cities.

Code:

```
public class A9_B1_14_practical2B {
    static final int INF = Integer.MAX_VALUE;



    public static int distance(int x1, int y1, int x2, int y2) {
        return (int) Math.round(Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2)));
    }


    public static void primMST(int cost[][], int n, String[] cities) {
        int[] near = new int[n];
        int[][] t = new int[n - 1][2];
        int mincost = 0;


        int k = -1, l = -1, min = INF;
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
```

```
        if (cost[i][j] < min) {
            min = cost[i][j];
            k = i;
            l = j;
        }
    }
}

mincost = cost[k][l];
t[0][0] = k;
t[0][1] = l;

for (int i = 0; i < n; i++) {
    if (cost[i][k] < cost[i][l])
        near[i] = k;
    else
        near[i] = l;
}

near[k] = near[l] = -1;

for (int i = 1; i < n - 1; i++) {
    int j = -1;
    min = INF;
    for (int v = 0; v < n; v++) {
        if (near[v] != -1 && cost[v][near[v]] < min) {
            min = cost[v][near[v]];
            j = v;
        }
    }
```

```
            t[i][0] = j;
            t[i][1] = near[j];
            mincost += cost[j][near[j]];
            near[j] = -1;

/////////////////////////////////////////////////////////////////////////////////
            for (int v = 0; v < n; v++) {
                if (near[v] != -1 && cost[v][j] < cost[v][near[v]]) {
                    near[v] = j;
                }
            }
        }

        System.out.println("Connections in MST:");
        for (int i = 0; i < n - 1; i++) {
            int u = t[i][0];
            int v = t[i][1];
            System.out.println(cities[u] + " - " + cities[v] + " : " + cost[u][v]);
        }
        System.out.println("Minimum leased line cost = " + mincost);
    }

    public static void main(String[] args) {
        String[] cities = {"C1", "C2", "C3", "C4", "C5"};
        int[][] coords = {
            {2, 3},
            {5, 4},
            {1, 7},
            {6, 1},
            {8, 5}
        };

        int n = cities.length;
        int[][] cost = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (i == j) cost[i][j] = INF;
                else cost[i][j] = distance(coords[i][0], coords[i][1],
                                           coords[j][0], coords[j][1]);
            }
        }

        primMST(cost, n, cities);
    }
}
```



```
PS D:\Abhishek\RBU\DAA\Practicals\Practical no 2>  &
-cp' 'C:\Users\USER\AppData\Roaming\Code\User\worksp
in' 'A9_B1_14_practical2B'
Connections in MST:
C1 - C2 : 3
C4 - C2 : 3
C5 - C2 : 3
C3 - C1 : 4
Minimum leased line cost = 13
PS D:\Abhishek\RBU\DAA\Practicals\Practical no 2>
```

Conclusion:

Hence, we successfully found MST using prims algorithm for the following problem statements.

Github: https://github.com/Shadow3456rh/DAA-Rbu-Practicals/tree/main/Practical%20no%202