PRACTICAL NO : 02

Name : Soham Adgokar

Roll No : A2-B2-28

Date : 18/08/25

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.

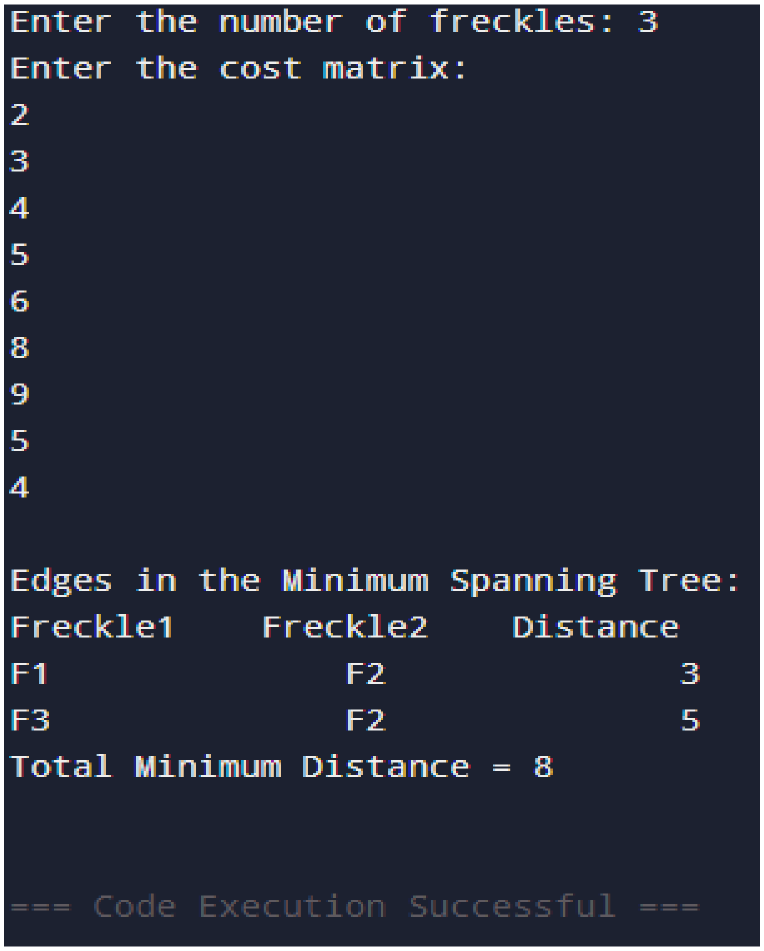
• Consider the distance between freckles as input in the form of matrix from the user.

• 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 :

```
#include <stdio.h>
#include <limits.h>

#define MAX 20

void primMST(int cost[MAX][MAX], int n) {
    int near[MAX], t[MAX][3];
    int total_cost = 0;


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

    t[0][0] = u;
    t[0][1] = v;
    t[0][2] = cost[u][v];
    total_cost += cost[u][v];


    for (int i = 0; i < n; i++) {
        if (cost[i][u] < cost[i][v])
            near[i] = u;
        else
            near[i] = v;
    }
    near[u] = near[v] = -1;
    for (int i = 1; i < n - 1; i++) {
        min = INT_MAX;
        int k = -1;
        for (int j = 0; j < n; j++) {
            if (near[j] != -1 && cost[j][near[j]] < min) {
                k = j;
                min = cost[j][near[j]];
            }
        }
        t[i][0] = k;
        t[i][1] = near[k];
        t[i][2] = cost[k][near[k]];
        total_cost += cost[k][near[k]];
        near[k] = -1;

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


    printf("\nEdges in the Minimum Spanning Tree:\n");
    printf("Freckle1\tFreckle2\tDistance\n");
    for (int i = 0; i < n - 1; i++) {
        printf("F%d\t\t\t\tF%d\t\t\t\t%d\n", t[i][0] + 1, t[i][1] + 1,
t[i][2]);
    }
    printf("Total Minimum Distance = %d\n", total_cost);
}

int main() {
    int n, cost[MAX][MAX];

    printf("Enter the number of freckles: ");
    scanf("%d", &n);
    printf("Enter the cost matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &cost[i][j]);
            if (i != j && cost[i][j] == 0)
                cost[i][j] = INT_MAX;
        }
    }

    primMST(cost, n);
    return 0;
}
```

Output :



```
Enter the number of freckles: 3
Enter the cost matrix:
2
3
4
5
6
8
9
5
4

Edges in the Minimum Spanning Tree:
Freckle1    Freckle2    Distance
F1          F2              3
F3          F2              5
Total Minimum Distance = 8



=== Code Execution Successful ===
```

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.

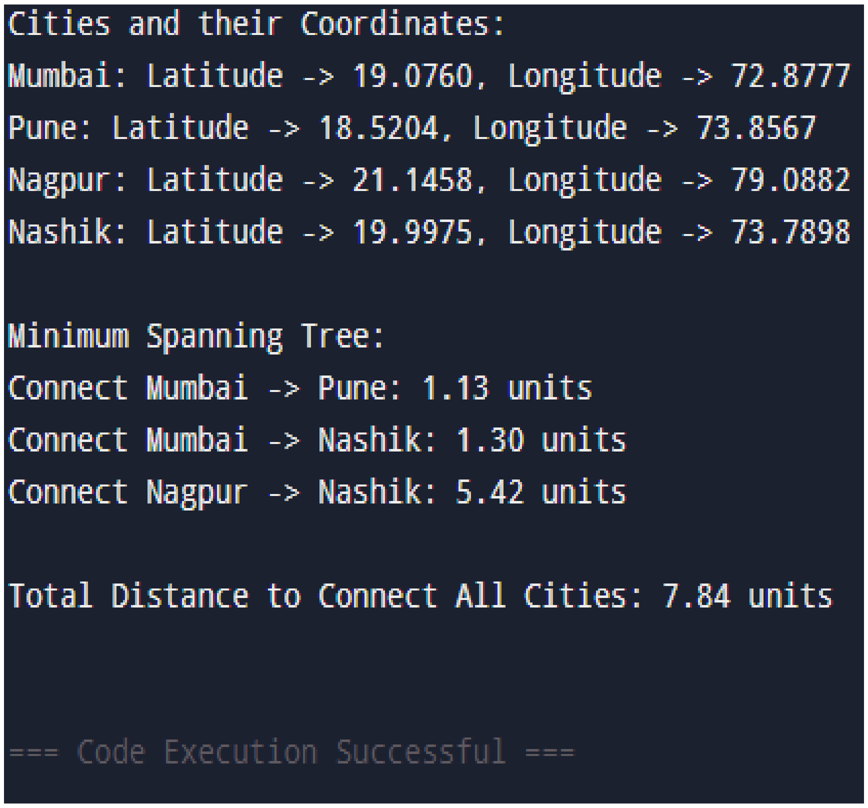
B. Repeat the above for cities in different states.

Code :

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define N 4
```

```
typedef struct {
    int u, v;
    double dist;
} Edge;

double cityCoords[N][2] = {
    {19.0760, 72.8777},
    {18.5204, 73.8567},
    {21.1458, 79.0882},
    {19.9975, 73.7898},
};

char* cityNames[N] = {
    "Mumbai", "Pune", "Nagpur", "Nashik"
};

double getDist(double x1, double y1, double x2, double y2) {
    return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
}

int find(int par[], int i) {
    if (par[i] != i)
        par[i] = find(par, par[i]);
    return par[i];
}

void join(int par[], int rank[], int x, int y) {
    int xr = find(par, x);
    int yr = find(par, y);
    if (xr != yr) {
        if (rank[xr] < rank[yr])
            par[xr] = yr;
        else if (rank[xr] > rank[yr])
            par[yr] = xr;
        else {
            par[yr] = xr;
            rank[xr]++;
        }
    }
}

int cmpEdges(const void* a, const void* b) {
    Edge* e1 = (Edge*)a;
    Edge* e2 = (Edge*)b;
    return (e1->dist > e2->dist) - (e1->dist < e2->dist);
}

int main() {
    Edge edges[N * (N - 1) / 2];
    int eCount = 0;

    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            double d = getDist(cityCoords[i][0], cityCoords[i][1],
                               cityCoords[j][0], cityCoords[j][1]);
            edges[eCount++] = (Edge){i, j, d};
        }
    }

    qsort(edges, eCount, sizeof(Edge), cmpEdges);

    int par[N], rank[N];
    for (int i = 0; i < N; i++) {
        par[i] = i;
        rank[i] = 0;
    }

    printf("Cities and their Coordinates:\n");
    for (int i = 0; i < N; i++)
        printf("%s: Latitude -> %.4f, Longitude -> %.4f\n",
cityNames[i], cityCoords[i][0], cityCoords[i][1]);

    printf("\nMinimum Spanning Tree:\n");
    double total = 0;
    int used = 0;

    for (int i = 0; i < eCount && used < N - 1; i++) {
        int u = edges[i].u;
        int v = edges[i].v;

        if (find(par, u) != find(par, v)) {
            printf("Connect %s -> %s: %.2f units\n", cityNames[u],
cityNames[v], edges[i].dist);
            total += edges[i].dist;
            join(par, rank, u, v);
            used++;
        }
    }

    printf("\nTotal Distance to Connect All Cities: %.2f units\n", total);
    return 0;
}
```

Output :



```
Cities and their Coordinates:
Mumbai: Latitude -> 19.0760, Longitude -> 72.8777
Pune: Latitude -> 18.5204, Longitude -> 73.8567
Nagpur: Latitude -> 21.1458, Longitude -> 79.0882
Nashik: Latitude -> 19.9975, Longitude -> 73.7898

Minimum Spanning Tree:
Connect Mumbai -> Pune: 1.13 units
Connect Mumbai -> Nashik: 1.30 units
Connect Nagpur -> Nashik: 5.42 units

Total Distance to Connect All Cities: 7.84 units



=== Code Execution Successful ===
```