RBU, Nagpur
CSE III Sem
PRACTICAL NO. 2

Name:	**Kishan Saini**
Sec-Batch-Rollno:	**A4-B4-55**

Part A:

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

#define V 4

int minKey(int key[], bool inMST[]) {
    int min = INT_MAX, min_index = -1;
    for (int v = 0; v < V; v++) {
        if (!inMST[v] && key[v] < min) {
            min = key[v];
            min_index = v;
        }
    }
    return min_index;
}

void primMST(int graph[V][V]){
    int parent[V];
    int key[V];
    bool inMST[V];
    int totalCost = 0;

    for(int i = 0; i <= V-1; i++){
        key[i] = INT_MAX;
        inMST[i] = false;
    }

    key[0] = 0;
    parent[0] = -1;

    for(int i = 0; i < V; i++){
        int u = minKey(key, inMST);
        if (u == -1) break;
```

```
            inMST[u] = true;
            for (int v = 0; v < V; v++){
                if(graph[u][v] != 0 && inMST[v] == false && graph[u][v]
< key[v]){
                    key[v] = graph[u][v];
                    parent[v] = u;
                }
            }

        }

        printf("Edges in MST:\n"); for (int i =
        1; i < V; i++) {
            printf("%d - %d   Weight: %d\n", parent[i], i, key[i]); totalCost += key[i];
        }

        printf("Total cost of Ink: %d", totalCost);

}

int main() {

        int graph[V][V] = {
            {0, 5, 8, 0},
            {5, 0, 10, 15},
            {8, 10, 0, 20},
            {0, 15, 20, 0}
        };

        primMST(graph);

        return 0;
}
```

OUTPUT:

```
Edges in MST:
0 - 1  Weight: 5
0 - 2  Weight: 8
1 - 3  Weight: 15
Total cost of Ink: 28
PS D:\Rana\College\DAA lab>
```

Part B:

```c
#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
PS D:\Rana\College\DAA lab>
```