ASSIGNMENT 4

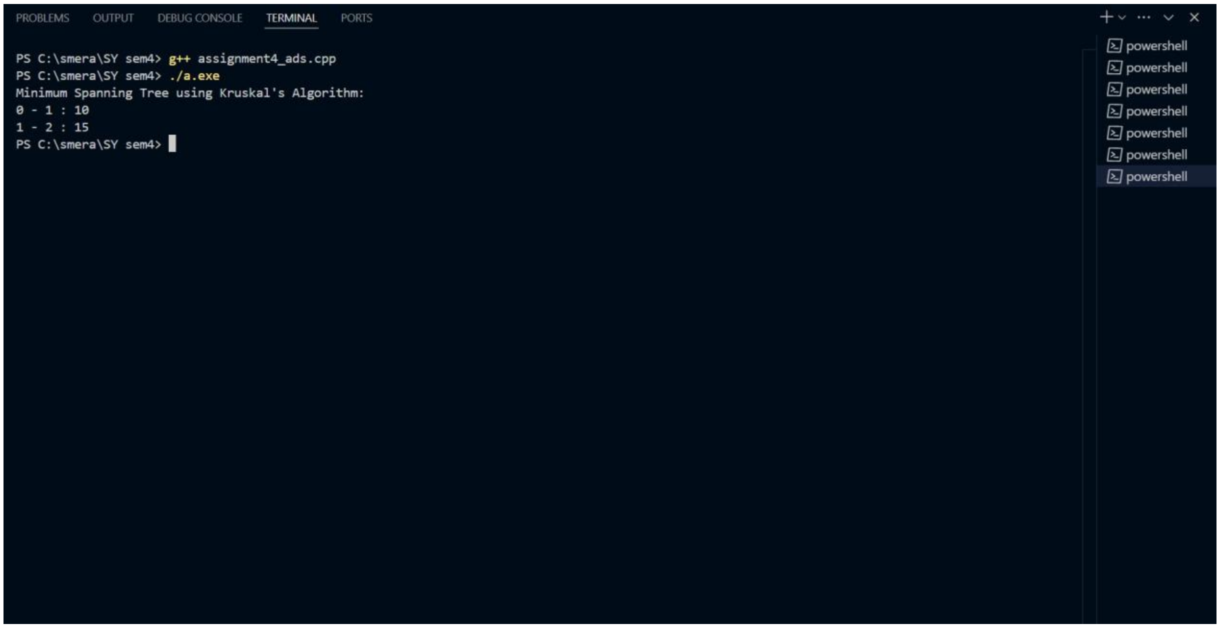
AIM:
Represent a graph of your college campus using adjacency list /adjacency matrix. Nodes should represent the various departments and links should represent the distance between them. Find a minimum spanning tree using Kruskal's algorithm or using Prim's algorithm.

SOURCE CODE:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

#define V 5 // Number of vertices

// Structure to represent an edge
struct Edge {
    int src, dest, weight;
};

// Disjoint Set (Union-Find) utility functions
int find(int parent[], int i) {
    if (parent[i] == i)
        return i;
    return parent[i] = find(parent, parent[i]); // Path compression
}

void unionSets(int parent[], int rank[], int x, int y) {
    int xroot = find(parent, x);
    int yroot = find(parent, y);

    if (rank[xroot] < rank[yroot])
        parent[xroot] = yroot;
    else if (rank[xroot] > rank[yroot])
        parent[yroot] = xroot;
```

```
    else {
        parent[yroot] = xroot;
        rank[xroot]++;
    }
}

// Kruskal's algorithm function
void kruskalMST(vector<Edge>& edges) {
    vector<Edge> result; // Stores the result MST
    int parent[V], rank[V];

    // Initially, each vertex is its own parent
    for (int i = 0; i < V; ++i) {
        parent[i] = i;
        rank[i] = 0;
    }

    // Sort all the edges in increasing order of their weight
    sort(edges.begin(), edges.end(), [](Edge a, Edge b) {
        return a.weight < b.weight;
    });

    int e = 0, i = 0;
    while (e < V - 1 && i < edges.size()) {
        Edge next = edges[i++];
        int x = find(parent, next.src);
        int y = find(parent, next.dest);

        if (x != y) {
            result.push_back(next);
            unionSets(parent, rank, x, y);
            e++;
        }
    }
```

```
    // Print MST
    cout << "Edge \tWeight\n";
    for (auto& edge : result)
        cout << edge.src << " - " << edge.dest << "\t" << edge.weight << endl;
}

int main() {
    int graph[V][V] = {
        {0, 2, 0, 6, 0},
        {2, 0, 3, 8, 5},
        {0, 3, 0, 0, 7},
        {6, 8, 0, 0, 9},
        {0, 5, 7, 9, 0}
    };

    vector<Edge> edges;

    // Extract edges from upper triangle of adjacency matrix (since undirected)
    for (int i = 0; i < V; i++) {
        for (int j = i + 1; j < V; j++) {
            if (graph[i][j] != 0) {
                edges.push_back({i, j, graph[i][j]});
            }
        }
    }

    kruskalMST(edges);

    return 0;
}
```
OUTPUT:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\smera\SY sem4> g++ assignment4_ads.cpp
PS C:\smera\SY sem4> ./a.exe
Minimum Spanning Tree using Kruskal's Algorithm:
0 - 1 : 10
1 - 2 : 15
PS C:\smera\SY sem4>
powershell
powershell
powershell
powershell
powershell
powershell
powershell

CONCLUSION:

The implementation of graph representation for a college campus and the application of Minimum Spanning Tree (MST) algorithms such as Kruskal’s or Prim’s help in optimizing campus connectivity. By efficiently connecting various departments with minimal total distance, this approach reduces infrastructure costs and ensures effective planning. The use of MST algorithms demonstrates how graph theory can be applied in real-world scenarios like network design, transportation planning, and resource management. This study highlights the importance of graph algorithms in solving complex connectivity problems efficiently.