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 <climits> // For INT\_MAX

using namespace std;

#define V 5 // Number of vertices

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}

};

// Function to find the vertex with minimum key value

int minKey(int key[], bool mstSet[]) {

int min = INT\_MAX, index;

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

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

min = key[v];

index = v;

}

}

return index;

}

// Function to construct and print MST using Prim's algorithm

void primMST() {

int parent[V]; // Array to store constructed MST

int key[V]; // Key values used to pick minimum weight edge

bool mstSet[V]; // To represent set of vertices included in MST

// Initialize all keys as infinite and mstSet[] as false

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

key[i] = INT\_MAX;

mstSet[i] = false;

}

// Start from first vertex

key[0] = 0; // Make key 0 so that this vertex is picked first

parent[0] = -1; // First node is always root of MST

// Construct MST

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

int u = minKey(key, mstSet); // Pick the minimum key vertex

mstSet[u] = true; // Include u in MST

// Update key and parent of the adjacent vertices

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

// Update only if graph[u][v] is non-zero, v is not in MST,

// and the edge weight is less than current key[v]

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

parent[v] = u;

key[v] = graph[u][v];

}

}

}

// Print the constructed MST

cout << "Edge \tWeight\n";

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

cout << parent[i] << " - " << i << "\t" << graph[i][parent[i]] << endl;

}

}

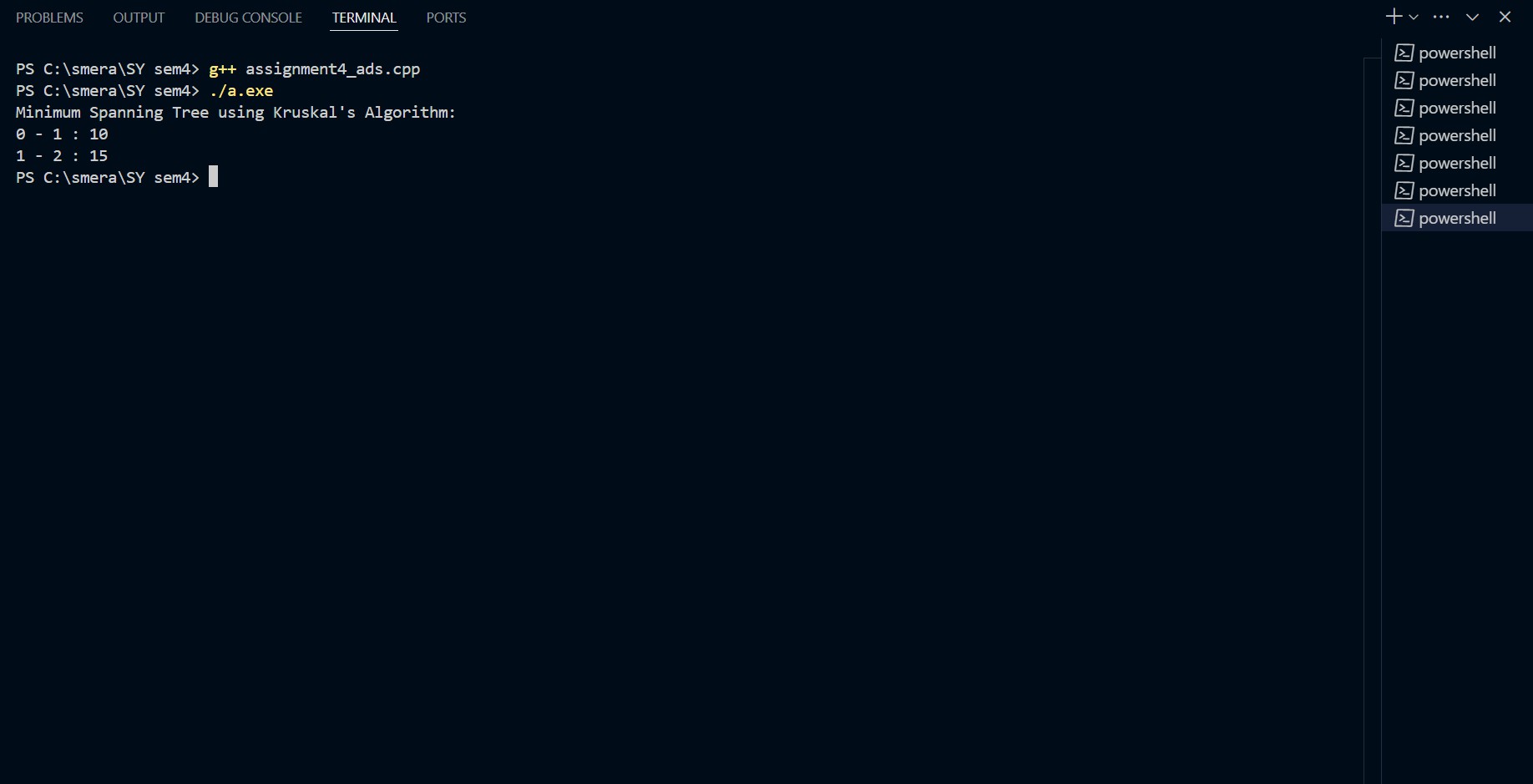
int main() {

primMST();

return 0;

}

# OUTPUT:

****

**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.