/\*

Problem Statement: You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

\*/

#include <iostream>

#include <vector>

#include <climits>

using namespace std;

void primsAlgorithm(vector<vector<int>> &graph, int V) {

vector<int> parent(V); // Stores the MST

vector<int> key(V, INT\_MAX); // Used to pick minimum weight edge

vector<bool> mstSet(V, false); // To represent set of vertices included in MST

key[0] = 0; // Start from vertex 0

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

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

// Find the vertex with the minimum key value

int u = -1;

int minKey = INT\_MAX;

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

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

minKey = key[v];

u = v;

}

}

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

// Update key and parent of adjacent vertices

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

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

parent[v] = u;

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

}

}

}

// Print the MST

cout << "Edge \tWeight\n";

int totalCost = 0;

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

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

totalCost += graph[i][parent[i]];

}

cout << "Total Minimum Cost: " << totalCost << endl;

}

int main() {

int V = 5; // Number of offices

// Adjacency matrix representation

vector<vector<int>> graph = {

{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}

};

primsAlgorithm(graph, V);

return 0;

}

/\*Edge Weight

0 - 1 2

1 - 2 3

0 - 3 6

1 - 4 5

Total Minimum Cost: 16 \*/