Assignment No. 9

Name – Manish Namdev Barage

PRN – 22520007

Batch – T7

Problem Statement – To minimum cost spanning tree of a given undirected graph using Prims Algorithm

Code –

```
#include <iostream>
#include<bits/stdc++.h>

using namespace std;

class Graph {
private:
    int numVertices;
    vector<vector<int>> adjMatrix;

public:
    Graph(int vertices) {
        numVertices = vertices;
        adjMatrix.resize(numVertices, vector<int>(numVertices, 0));
    }

    void addEdge(int src, int dest, int weight) {
        // Adding edges for an undirected graph
        adjMatrix[src][dest] = weight;
        adjMatrix[dest][src] = weight;
    }

    void primMST() {
        vector<bool> inMST(numVertices, false);
        vector<int> key(numVertices, INT_MAX);
        vector<int> parent(numVertices, -1);
        priority_queue<pair<int, int>, vector<pair<int, int>>,
greater<pair<int, int>>> pq;

        int src = 0; // Start from vertex 0
        pq.push(make_pair(0, src));
        key[src] = 0;

        while (!pq.empty()) {
            int u = pq.top().second;
            pq.pop();
            inMST[u] = true;

            // Iterate through all adjacent vertices of u
            for (int v = 0; v < numVertices; ++v) {
```

```
                if (adjMatrix[u][v] != 0 && !inMST[v] &&
adjMatrix[u][v] < key[v]) {
                    key[v] = adjMatrix[u][v];
                    pq.push(make_pair(key[v], v));
                    parent[v] = u;
                }
            }
        }

        int cost =0;

        // Print the MST edges
        cout << "Edge \tWeight\n";
        for (int i = 1; i < numVertices; ++i) {
            cout << parent[i] << " - " << i << "\t" <<
adjMatrix[i][parent[i]] << endl;
            cost +=adjMatrix[i][parent[i]];
        }

        cout<<"And total cost of MST is : "<<cost<<endl;
    }

    void printGraph() {
        for (int i = 0; i < numVertices; ++i) {
            cout << "Vertex " << i << " is connected to:\n";
            for (int j = 0; j < numVertices; ++j) {
                if (adjMatrix[i][j] != 0) {
                    cout << "  Vertex " << j << " with weight " <<
adjMatrix[i][j] << endl;
                }
            }
            cout << "---------------\n";
        }
    }
};

int main() {
    int numVertices = 5; // Change this to the number of vertices in
your graph
    Graph graph(numVertices);

    // Adding edges to the graph
    graph.addEdge(0, 1, 2);
    graph.addEdge(0, 3, 6);
    graph.addEdge(1, 2, 3);
    graph.addEdge(1, 3, 8);
    graph.addEdge(1, 4, 5);
    graph.addEdge(2, 4, 7);
    graph.addEdge(3, 4, 9);

    // Print the original graph
    cout << "Original Graph:\n";
    graph.printGraph();
```

```
    // Find and print the Minimum Spanning Tree (MST) using Prim's
algorithm
    cout << "\nMinimum Spanning Tree (MST) using Prim's Algorithm:\n";
    graph.primMST();

    return 0;
}
```