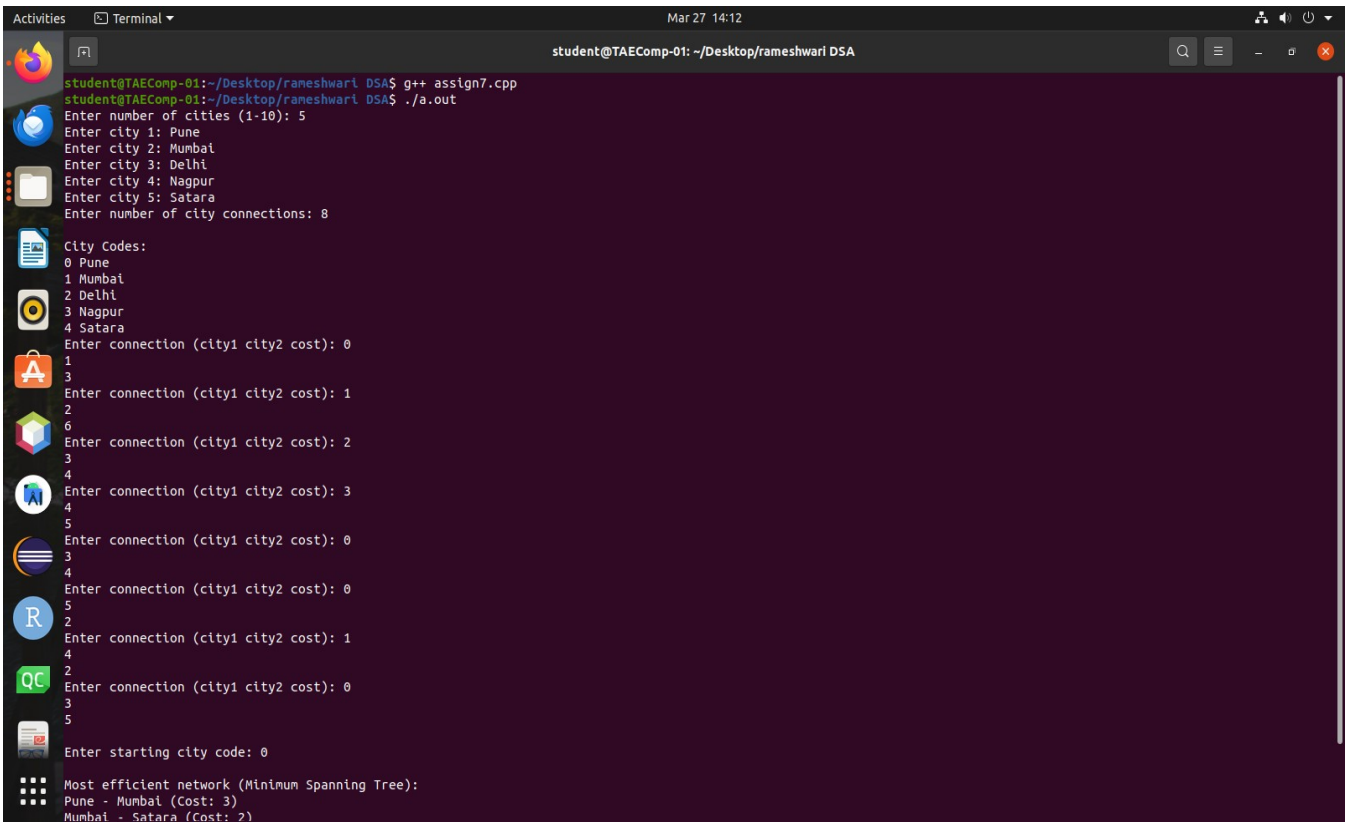
Assignment 7

code:

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

#define MAX 10 // Maximum number of cities

// Structure for edges
struct Edge {
    int start, end, weight;
};

class Graph {
    int adj[MAX][MAX];        // Adjacency matrix
    string city[MAX];         // City names
    int numCities;            // Number of cities
    vector<Edge> mst;         // Minimum Spanning Tree (MST)
    int totalCost;            // Total cost of MST

public:
    Graph();                  // Constructor
    void prims(int start);    // Prim's algorithm to find MST
    void showMST();           // Display the MST
};

// Constructor: Initializes the graph
Graph::Graph() {
    totalCost = 0;
    cout << "Enter number of cities (1-" << MAX << "): ";
    cin >> numCities;

    numCities = min(numCities, MAX);

    // Get city names
    for (int i = 0; i < numCities; i++) {
        cout << "Enter city " << i + 1 << ": ";
        cin >> city[i];
    }

    // Initialize adjacency matrix with "infinity" (no direct connections)
    for (int i = 0; i < numCities; i++) {
        for (int j = 0; j < numCities; j++) {
            adj[i][j] = INT_MAX;  // Initialize with large value
        }
    }

    // Get city connections (edges)
    int connections;
    cout << "Enter number of city connections: ";
    cin >> connections;

    // Show city codes
    cout << "\nCity Codes: \n";
    for (int i = 0; i < numCities; i++) {
        cout << i << " " << city[i] << endl;
    }

    // Read edges
    for (int i = 0; i < connections; i++) {
```

```
        int x, y, cost;
        cout << "Enter connection (city1 city2 cost): ";
        cin >> x >> y >> cost;
        adj[x][y] = cost;  // Undirected graph
        adj[y][x] = cost;  // Symmetric
    }
}

// Prim's Algorithm to find MST
void Graph::prims(int start) {
    bool visited[MAX] = {false};  // Array to track visited cities
    visited[start] = true;        // Mark the starting city as visited

    while (mst.size() < numCities - 1) {  // While there are still edges to add
        Edge minEdge = {0, 0, INT_MAX};  // Initialize minEdge with maximum cost

        // Find the smallest edge from any visited city
        for (int i = 0; i < numCities; i++) {
            if (visited[i]) {  // If city 'i' is visited
                for (int j = 0; j < numCities; j++) {
                    if (!visited[j] && adj[i][j] < minEdge.weight) {  // If city 'j' is unvisited and edge weight is less
                        minEdge = {i, j, adj[i][j]};  // Update minEdge
                    }
                }
            }
        }

        // Add the edge to MST
        mst.push_back(minEdge);
        totalCost += minEdge.weight;
        visited[minEdge.end] = true;  // Mark the destination city as visited
    }
}

// Display the Minimum Spanning Tree (MST)
void Graph::showMST() {
    cout << "\nMost efficient network (Minimum Spanning Tree):\n";
    for (Edge e : mst) {
        cout << city[e.start] << " - " << city[e.end] << " (Cost: " << e.weight << ")\n";
    }
    cout << "Total network cost: " << totalCost << endl;
}

// Main function
int main() {
    Graph g;
    int start;
    cout << "\nEnter starting city code: ";
    cin >> start;

    g.prims(start);
    g.showMST();

    return 0;
}
```

Output:


Activities
Terminal
Mar 27 14:12
student@TAEComp-01: ~/Desktop/rameshwari DSA
student@TAEComp-01:~/Desktop/rameshwari DSA$ g++ assign7.cpp
student@TAEComp-01:~/Desktop/rameshwari DSA$ ./a.out
Enter number of cities (1-10): 5
Enter city 1: Pune
Enter city 2: Mumbai
Enter city 3: Delhi
Enter city 4: Nagpur
Enter city 5: Satara
Enter number of city connections: 8
City Codes:
0 Pune
1 Mumbai
2 Delhi
3 Nagpur
4 Satara
Enter connection (city1 city2 cost): 0
1
3
Enter connection (city1 city2 cost): 1
2
6
Enter connection (city1 city2 cost): 2
3
4
Enter connection (city1 city2 cost): 3
4
5
Enter connection (city1 city2 cost): 0
3
4
Enter connection (city1 city2 cost): 0
5
2
Enter connection (city1 city2 cost): 1
4
2
Enter connection (city1 city2 cost): 0
3
5
Enter starting city code: 0
Most efficient network (Minimum Spanning Tree):
Pune - Mumbai (Cost: 3)
Mumbai - Satara (Cost: 2)