Assignment No. 7

Name: Yogesh Giridhar Chimandare

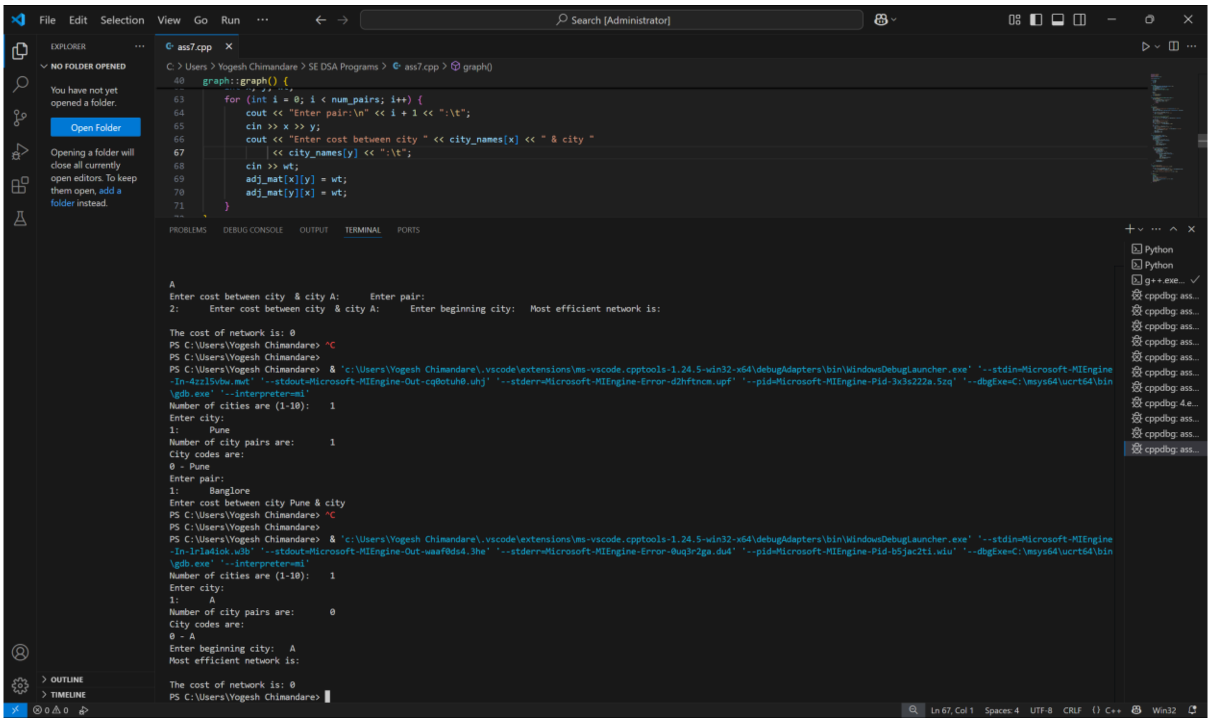
Roll No: COA218

Programme:

```
#include <iostream>
#include <vector>
#include <cstdint>
#define MAX_NUM_CITIES 10

using namespace std;

struct edge {
   int start;
   int end;
   int wt;
};

class graph {
   int adj_mat[MAX_NUM_CITIES][MAX_NUM_CITIES] = {0};
   string city_names[MAX_NUM_CITIES];
   int city_count;
   edge mst[MAX_NUM_CITIES - 1];
   void add_to_list(vector<edge> &, edge);
   int cost;

  public:
   graph();

   void prims_algo(int);
```

```
    void display_mst();
};

void graph::add_to_list(vector<edge> &list, edge e) {
    list.push_back(e);
    for (int i = list.size() - 1; i > 0; i--) {
        if (list[i].wt < list[i - 1].wt) {
            swap(list[i], list[i - 1]);
        } else {
            break;
        }
    }
}

graph::graph() {
    cost = 0;
    cout << "Number of cities are (1-" << MAX_NUM_CITIES << "):\t";
    cin >> city_count;
    city_count = (city_count > MAX_NUM_CITIES) ? MAX_NUM_CITIES : city_count;

    for (int i = 0; i < city_count; i++) {
        cout << "Enter city:\n" << i + 1 << ":\t";
        cin >> city_names[i];
    }

    for (int i = 0; i < city_count; i++)
        for (int j = 0; j < city_count; j++) adj_mat[i][j] = INT32_MAX;

    int num_pairs;
    cout << "Number of city pairs are:\t";
    cin >> num_pairs;
    cout << "City codes are:\t" << endl;
    for (int i = 0; i < city_count; i++) {
        cout << i << " - " << city_names[i] << endl;
    }
    int x, y, wt;
    for (int i = 0; i < num_pairs; i++) {
        cout << "Enter pair:\n" << i + 1 << ":\t";
        cin >> x >> y;
        cout << "Enter cost between city " << city_names[x] << " & city "
             << city_names[y] << ":\t";
        cin >> wt;
        adj_mat[x][y] = wt;
        adj_mat[y][x] = wt;
    }
}

void graph::prims_algo(int start) {
    bool visited[MAX_NUM_CITIES] = {0};
    int visited_count = 1;
    visited[start] = 1;
    vector<edge> adj;
    for (int i = 0; i < city_count; i++) {
        if (adj_mat[start][i] != INT32_MAX) {
            edge e;
            e.start = start;
            e.end = i;
            e.wt = adj_mat[start][i];
            add_to_list(adj, e);
        }
    }

    while (visited_count != city_count) {
        edge m = adj.front();
        adj.erase(adj.begin());
        if (!visited[m.end]) {
            mst[visited_count - 1] = m;
            cost += m.wt;
            for (int i = 0; i < city_count; i++) {
                if (adj_mat[m.end][i] != INT32_MAX) {
                    edge e;
                    e.start = m.end;
                    e.end = i;
                    e.wt = adj_mat[e.start][i];
                    add_to_list(adj, e);
                }
            }
            visited[m.end] = 1;
            visited_count++;
        }
    }
}

void graph::display_mst() {
    cout << "Most efficient network is:\t" << endl;

    for (int i = 0; i < city_count - 1; i++) {
        cout << city_names[mst[i].start] << " to " << city_names[mst[i].end]
             << " of weight " << mst[i].wt << endl;
    }
    cout << endl << "The cost of network is:\t" << cost << endl;
}

int main() {
    graph g;
    int start;
    cout << "Enter beginning city:\t";
    cin >> start;
    start = (start > MAX_NUM_CITIES - 1) ? 0 : start;
    g.prims_algo(start);
    g.display_mst();
    return 0;
}
```

Output:



```
graph::graph() {
        for (int i = 0; i < num_pairs; i++) {
            cout << "Enter pair:\n" << i + 1 << ":\t";
            cin >> x >> y;
            cout << "Enter cost between city " << city_names[x] << " & city "
                 << city_names[y] << ":\t";
            cin >> wt;
            adj_mat[x][y] = wt;
            adj_mat[y][x] = wt;
        }
```

```
A
Enter cost between city  & city A:      Enter pair:
2:      Enter cost between city  & city A:      Enter beginning city:   Most efficient network is:

The cost of network is: 0
PS C:\Users\Yogesh Chimandare> ^C
PS C:\Users\Yogesh Chimandare>
PS C:\Users\Yogesh Chimandare>  & 'c:\Users\Yogesh Chimandare\.vscode\extensions\ms-vscode.cpptools-1.24.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine
-In-4zzl5vbw.mwt' '--stdout=Microsoft-MIEngine-Out-cq0otuh0.uhj' '--stderr=Microsoft-MIEngine-Error-d2hftncm.upf' '--pid=Microsoft-MIEngine-Pid-3x3s222a.5zq' '--dbgExe=C:\msys64\ucrt64\bin
\gdb.exe' '--interpreter=mi'
Number of cities are (1-10):    1
Enter city:
1:      Pune
Number of city pairs are:       1
City codes are:
0 - Pune
Enter pair:
1:      Banglore
Enter cost between city Pune & city
PS C:\Users\Yogesh Chimandare> ^C
PS C:\Users\Yogesh Chimandare>
PS C:\Users\Yogesh Chimandare>  & 'c:\Users\Yogesh Chimandare\.vscode\extensions\ms-vscode.cpptools-1.24.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine
-In-1rla4iok.w3b' '--stdout=Microsoft-MIEngine-Out-waaf0ds4.3he' '--stderr=Microsoft-MIEngine-Error-0uq3r2ga.du4' '--pid=Microsoft-MIEngine-Pid-b5jac2ti.wiu' '--dbgExe=C:\msys64\ucrt64\bin
\gdb.exe' '--interpreter=mi'
Number of cities are (1-10):    1
Enter city:
1:      A
Number of city pairs are:       0
City codes are:
0 - A
Enter beginning city:   A
Most efficient network is:

The cost of network is: 0
PS C:\Users\Yogesh Chimandare>
```