```
#include <iostream>
#include <iomanip>
#include <vector>
#include <limits>
using namespace std;

class Graph {
public:
    int n;
    vector<vector<int>> mat;
    vector<string> offices;

    Graph(int no) {
        n = no;
        mat.resize(n, vector<int>(n, 0));
        offices.resize(n);
    }

    void create();
    void display();
    void minCost();
};

void Graph::create() {
    for (int i = 0; i < n; i++) {
        cout << "Enter name of office #" << (i + 1) << ": ";
        cin >> offices[i];
    }

    cout << "\nEnter cost to connect offices (Enter 0 if no direct connection):\n";
    for (int i = 0; i < n; i++) {
```

```
        for (int j = i + 1; j < n; j++) {
            cout << "Enter cost to connect " << offices[i] << " and " << offices[j] << ": ";
            cin >> mat[i][j];
            mat[j][i] = mat[i][j];
        }
    }
}

void Graph::display() {
    cout << "\nConnection Cost Matrix:\n\t";
    for (int i = 0; i < n; i++)
        cout << setw(10) << offices[i];
    cout << endl;

    for (int i = 0; i < n; i++) {
        cout << offices[i] << "\t";
        for (int j = 0; j < n; j++) {
            cout << setw(10) << mat[i][j];
        }
        cout << endl;
    }
}

void Graph::minCost() {
    vector<int> parent(n, -1);
    vector<int> key(n, numeric_limits<int>::max());
    vector<bool> inMST(n, false);

    key[0] = 0;
    parent[0] = -1;
```

```
    for (int count = 0; count < n - 1; count++) {
        int minKey = numeric_limits<int>::max(), u = -1;
        for (int v = 0; v < n; v++) {
            if (!inMST[v] && key[v] < minKey) {
                minKey = key[v];
                u = v;
            }
        }

        inMST[u] = true;

        for (int v = 0; v < n; v++) {
            if (mat[u][v] && !inMST[v] && mat[u][v] < key[v]) {
                parent[v] = u;
                key[v] = mat[u][v];
            }
        }
    }

    cout << "\nMinimum Cost to Connect Offices:\n";
    int totalCost = 0;
    for (int i = 1; i < n; i++) {
        cout << offices[parent[i]] << " - " << offices[i]
             << " : " << mat[i][parent[i]] << endl;
        totalCost += mat[i][parent[i]];
    }
    cout << "Total Minimum Cost: " << totalCost << endl;
}

int main() {
    int no;
```

```
    cout << "Enter number of offices: ";
    cin >> no;

    Graph telNet(no);
   cout << "\n# Creating network to connect offices:\n";
    telNet.create();

    int choice;
    do {
        cout << "\n# Menu:\n";
        cout << "1. Display Cost Chart\n";
        cout << "2. Find Minimum Cost of Connection\n";
        cout << "3. Exit\n";
        cout << "Select option: ";
        cin >> choice;
        switch (choice) {
            case 1:
                telNet.display();
                break;
            case 2:
                telNet.minCost();
                break;
            case 3:
                cout << "Exiting...\n";
                return 0;
            default:
                cout << "Invalid choice! Please try again.\n";
        }
    } while (true);
  return 0;
}
```

Enter number of offices: 3

Creating network to connect offices:

Enter name of office #1: A

Enter name of office #2: B

Enter name of office #3: C

Enter cost to connect offices (Enter 0 if no direct connection):

Enter cost to connect A and B: 10

Enter cost to connect A and C: 15

Enter cost to connect B and C: 5

Menu:

1. Display Cost Chart

2. Find Minimum Cost of Connection

3. Exit

Select option: 1

Connection Cost Matrix:

	A	B	C
A	0	10	15
B	10	0	5
C	15	5	0

Menu:

1. Display Cost Chart

2. Find Minimum Cost of Connection

3. Exit

Select option: 2

Minimum Cost to Connect Offices:

B - C : 5

A - B : 10

Total Minimum Cost: 15