```
#include <iostream>
#include <string>
using namespace std;
#define INF 99999

int adj_mat[50][50] = {0};
int n;
string offices[50];

void insert() {
    cout << "Enter the number of offices: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter office " << i + 1 << ": ";
        cin >> offices[i];
    }
    cout << "Enter the cost between the offices:\n";
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            cout << "Enter the cost between " << offices[i] << " and " << offices[j] << ": ";
            cin >> adj_mat[i][j];
            if (adj_mat[i][j] == 0 && i != j) {
                adj_mat[i][j] = INF;
            }
            adj_mat[j][i] = adj_mat[i][j];
        }
    }
}

void display() {
    cout << "Adjacency Matrix (Office Connection Costs):\n";
    cout << "    ";
    for (int i = 0; i < n; i++) {
        cout << offices[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < n; i++) {
        cout << offices[i] << " ";
        for (int j = 0; j < n; j++) {
            if (adj_mat[i][j] == INF) {
                cout << "INF ";
            } else {
                cout << adj_mat[i][j] << " ";
            }
        }
        cout << endl;
    }
}
```

```
int findMinVertex(int key[], bool visited[]) {
    int minIndex = -1, minValue = INF;
    for (int i = 0; i < n; i++) {
        if (!visited[i] && key[i] < minValue) {
            minValue = key[i];
            minIndex = i;
        }
    }
    return minIndex;
}


void primMST() {
    int parent[50];
    int key[50];
    bool visited[50] = {false};

    for (int i = 0; i < n; i++) {
        key[i] = INF;
    }
    key[0] = 0;
    parent[0] = -1;

    for (int count = 0; count < n - 1; count++) {
        int u = findMinVertex(key, visited);
        visited[u] = true;

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

    cout << "Minimum Spanning Tree:\n";
    int totalCost = 0;
    for (int i = 1; i < n; i++) {
        cout << offices[parent[i]] << " - " << offices[i] << " : " << adj_mat[i][parent[i]] << endl;
        totalCost += adj_mat[i][parent[i]];
    }
    cout << "Total cost of connecting all offices: " << totalCost << endl;
}

int main() {
    insert();
    display();
    primMST();
    return 0;
}
```