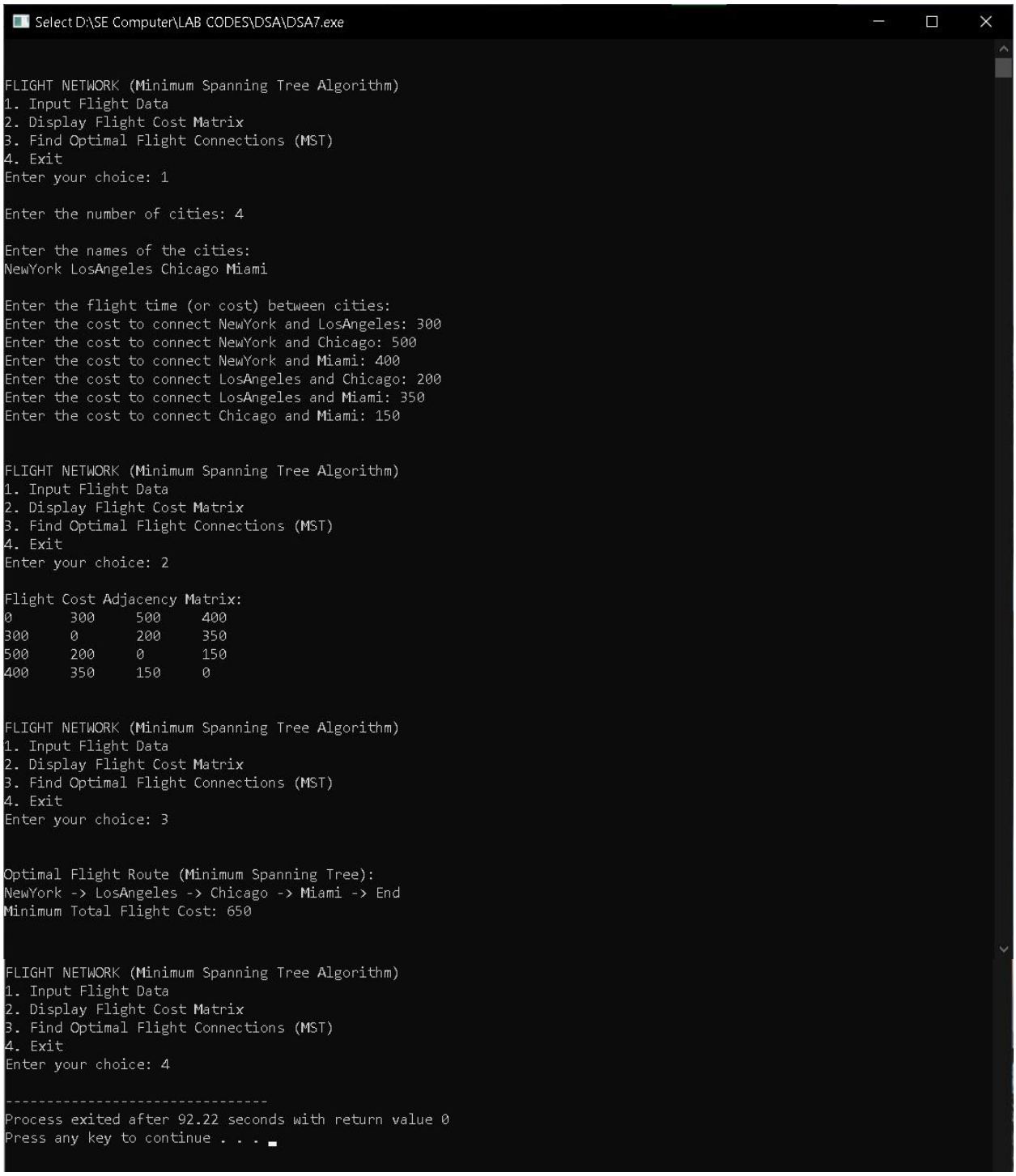
```cpp
#include <iostream>
#include <limits.h>
using namespace std;

class FlightNetwork {
    int n;  // Number of cities
    int adjacent[10][10];  // Adjacency matrix for flight costs
    string city[10];  // City names

public:
    void input();
    void display();
    void Prims();
};

// Function to input flight details
void FlightNetwork::input() {
    cout << "\nEnter the number of cities: ";
    cin >> n;

    cout << "\nEnter the names of the cities:\n";
    for (int i = 0; i < n; i++)
        cin >> city[i];

    cout << "\nEnter the flight time (or cost) between cities:\n";
    for (int i = 0; i < n; i++)
        for (int j = i; j < n; j++) {
            if (i == j) {
                adjacent[i][j] = 0;  // No self-loop
                continue;
            }
            cout << "Enter the cost to connect " << city[i] << " and " << city[j] << ": ";
            cin >> adjacent[i][j];
            adjacent[j][i] = adjacent[i][j];  // Undirected graph (bi-directional flights)
        }
}

// Function to display the adjacency matrix
void FlightNetwork::display() {
    cout << "\nFlight Cost Adjacency Matrix:\n";
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            cout << adjacent[i][j] << "\t";
        cout << "\n";
    }
}

// Function to find the Minimum Spanning Tree using Prim's Algorithm
void FlightNetwork::Prims() {
    int visit[n], minCost = 0, count = n - 1, minIndex, cost = 0;

    for (int i = 0; i < n; i++)
        visit[i] = 0;
```

```
54
55      cout << "\n\nOptimal Flight Route (Minimum Spanning Tree):\n";
56      visit[0] = 1;
57      cout << city[0] << " -> ";
58
59      while (count--) {
60          minCost = INT_MAX;
61
62          for (int i = 0; i < n; i++) {
63              for (int j = 0; j < n; j++) {
64                  if (visit[i] == 1 && adjacent[i][j] != 0 && adjacent[i][j] < minCost
    && visit[j] == 0) {
65                      minCost = adjacent[i][j];
66                      minIndex = j;
67                  }
68              }
69          }
70
71          visit[minIndex] = 1;
72          cout << city[minIndex] << " -> ";
73          cost += minCost;
74      }
75
76      cout << "End\n";
77      cout << "Minimum Total Flight Cost: " << cost << "\n";
78  }
79
80  // Main function
81  int main() {
82      FlightNetwork network;
83      int choice;
84
85      do {
86          cout << "\n\nFLIGHT NETWORK (Minimum Spanning Tree Algorithm)";
87          cout << "\n1. Input Flight Data";
88          cout << "\n2. Display Flight Cost Matrix";
89          cout << "\n3. Find Optimal Flight Connections (MST)";
90          cout << "\n4. Exit";
91          cout << "\nEnter your choice: ";
92          cin >> choice;
93
94          switch (choice) {
95              case 1:
96                  network.input();
97                  break;
98              case 2:
99                  network.display();
100                 break;
101             case 3:
102                 network.Prims();
103                 break;
104         }
105     } while (choice != 4);
106
107     return 0;
108 }
109
```

OutPut :


Select D:\SE Computer\LAB CODES\DSA\DSA7.exe
FLIGHT NETWORK (Minimum Spanning Tree Algorithm)
1. Input Flight Data
2. Display Flight Cost Matrix
3. Find Optimal Flight Connections (MST)
4. Exit
Enter your choice: 1
Enter the number of cities: 4
Enter the names of the cities:
NewYork LosAngeles Chicago Miami
Enter the flight time (or cost) between cities:
Enter the cost to connect NewYork and LosAngeles: 300
Enter the cost to connect NewYork and Chicago: 500
Enter the cost to connect NewYork and Miami: 400
Enter the cost to connect LosAngeles and Chicago: 200
Enter the cost to connect LosAngeles and Miami: 350
Enter the cost to connect Chicago and Miami: 150
FLIGHT NETWORK (Minimum Spanning Tree Algorithm)
1. Input Flight Data
2. Display Flight Cost Matrix
3. Find Optimal Flight Connections (MST)
4. Exit
Enter your choice: 2
Flight Cost Adjacency Matrix:
0 300 500 400
300 0 200 350
500 200 0 150
400 350 150 0
FLIGHT NETWORK (Minimum Spanning Tree Algorithm)
1. Input Flight Data
2. Display Flight Cost Matrix
3. Find Optimal Flight Connections (MST)
4. Exit
Enter your choice: 3
Optimal Flight Route (Minimum Spanning Tree):
NewYork -> LosAngeles -> Chicago -> Miami -> End
Minimum Total Flight Cost: 650
FLIGHT NETWORK (Minimum Spanning Tree Algorithm)
1. Input Flight Data
2. Display Flight Cost Matrix
3. Find Optimal Flight Connections (MST)
4. Exit
Enter your choice: 4
Process exited after 92.22 seconds with return value 0
Press any key to continue . . .