Assignment 7

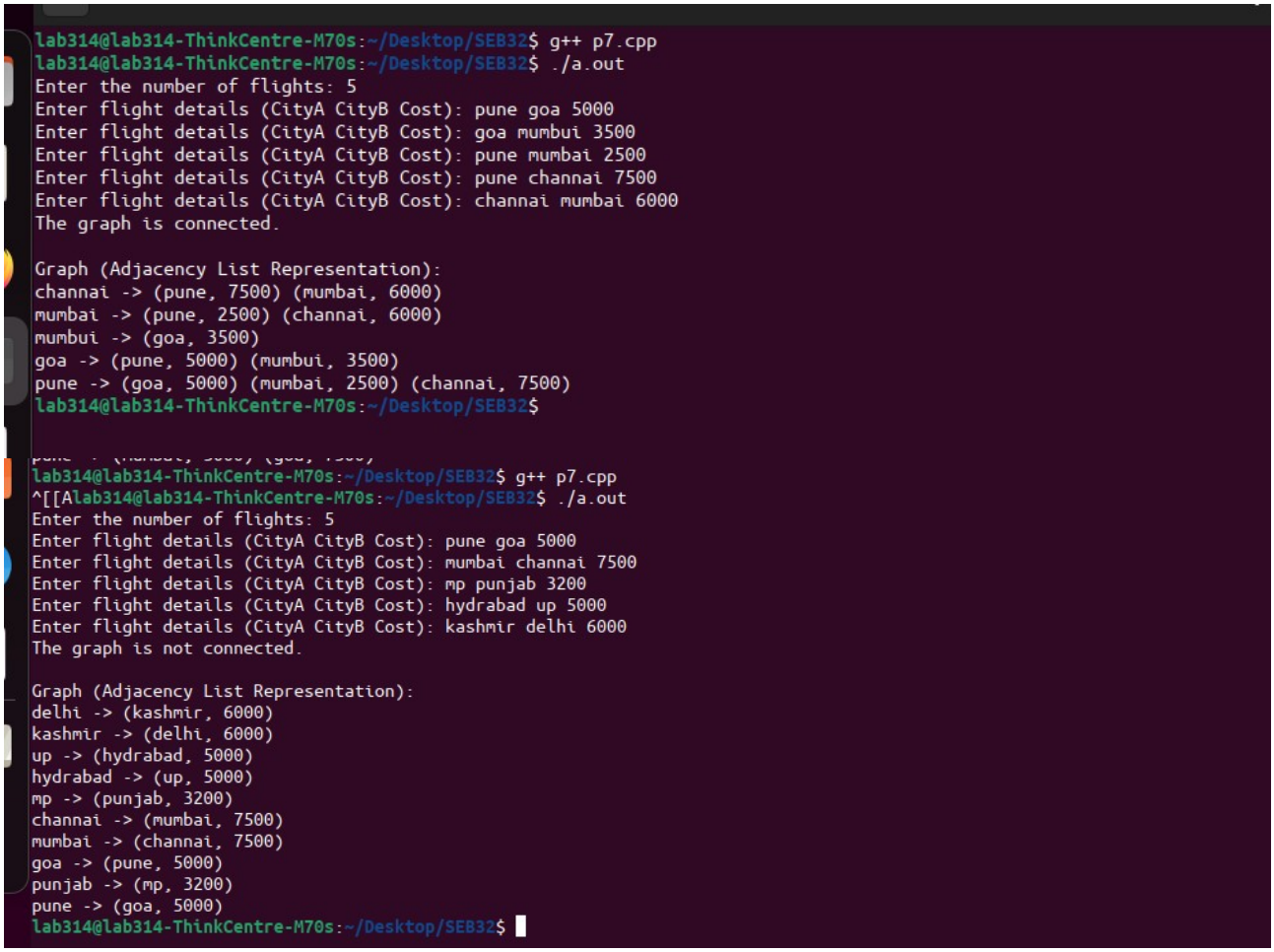
Problem Statement:-
There are flight paths between cities. If there is a flight between city A and city B then there is an edge between the cities. The cost of the edge can be the time that flight take to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node can be represented by airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Check whether the graph is connected or not. Justify the storage representation used.

Source Code:-

```
#include <iostream>
#include <unordered_map>
#include <list>
#include <vector>
#include <queue>
using namespace std;

class Graph {
private:
   unordered_map<string, list<pair<string, int>>> adjList;

public:

   void addFlight(const string& cityA, const string& cityB, int cost) {
      adjList[cityA].push_back({cityB, cost});
      adjList[cityB].push_back({cityA, cost});
   }


   bool isConnected() {
      unordered_map<string, bool> visited;
      if (adjList.empty()) return true;


      string startCity = adjList.begin()->first;


      queue<string> q;
      q.push(startCity);
      visited[startCity] = true;

      while (!q.empty()) {
         string city = q.front();
         q.pop();


         for (auto neighbor : adjList[city]) {
            if (!visited[neighbor.first]) {
               visited[neighbor.first] = true;
               q.push(neighbor.first);
```

```
            }
        }
    }


    return visited.size() == adjList.size();
}


void printGraph() {
    for (auto& pair : adjList) {
        cout << pair.first << " -> ";
        for (auto& neighbor : pair.second) {
            cout << "(" << neighbor.first << ", " << neighbor.second << ") ";
        }
        cout << endl;
    }
}
};

int main() {
    Graph g;
    int n;
    cout << "Enter the number of flights: ";
    cin >> n;


    for (int i = 0; i < n; ++i) {
        string cityA, cityB;
        int cost;
        cout << "Enter flight details (CityA CityB Cost): ";
        cin >> cityA >> cityB >> cost;
        g.addFlight(cityA, cityB, cost);
    }


    if (g.isConnected()) {
        cout << "The graph is connected." << endl;
    } else {
        cout << "The graph is not connected." << endl;
    }


    cout << "\nGraph (Adjacency List Representation):\n";
    g.printGraph();

    return 0;
}
```

Output:-


lab314@lab314-ThinkCentre-M70s:~/Desktop/SEB32$ g++ p7.cpp
lab314@lab314-ThinkCentre-M70s:~/Desktop/SEB32$ ./a.out
Enter the number of flights: 5
Enter flight details (CityA CityB Cost): pune goa 5000
Enter flight details (CityA CityB Cost): goa mumbui 3500
Enter flight details (CityA CityB Cost): pune mumbai 2500
Enter flight details (CityA CityB Cost): pune channai 7500
Enter flight details (CityA CityB Cost): channai mumbai 6000
The graph is connected.
Graph (Adjacency List Representation):
channai -> (pune, 7500) (mumbai, 6000)
mumbai -> (pune, 2500) (channai, 6000)
mumbui -> (goa, 3500)
goa -> (pune, 5000) (mumbui, 3500)
pune -> (goa, 5000) (mumbai, 2500) (channai, 7500)
lab314@lab314-ThinkCentre-M70s:~/Desktop/SEB32$
lab314@lab314-ThinkCentre-M70s:~/Desktop/SEB32$ g++ p7.cpp
^[[Alab314@lab314-ThinkCentre-M70s:~/Desktop/SEB32$ ./a.out
Enter the number of flights: 5
Enter flight details (CityA CityB Cost): pune goa 5000
Enter flight details (CityA CityB Cost): mumbai channai 7500
Enter flight details (CityA CityB Cost): mp punjab 3200
Enter flight details (CityA CityB Cost): hydrabad up 5000
Enter flight details (CityA CityB Cost): kashmir delhi 6000
The graph is not connected.
Graph (Adjacency List Representation):
delhi -> (kashmir, 6000)
kashmir -> (delhi, 6000)
up -> (hydrabad, 5000)
hydrabad -> (up, 5000)
mp -> (punjab, 3200)
channai -> (mumbai, 7500)
mumbai -> (channai, 7500)
goa -> (pune, 5000)
punjab -> (mp, 3200)
pune -> (goa, 5000)
lab314@lab314-ThinkCentre-M70s:~/Desktop/SEB32$