#include <iostream>

#include <vector>

#include <unordered\_map>

#include <unordered\_set>

#include <queue>

using namespace std;

class Graph{

private :

unordered\_map<string , vector<pair<string, int > > > adjList;

public :

void addFlight(string source , string destination , int cost){

adjList[source].push\_back({destination,cost});

adjList[destination].push\_back({source,cost});

}

bool isConnected(){

unordered\_set<string> visited;

queue<string> q;

q.push(adjList.begin() -> first); //start from any airport

visited.insert(adjList.begin() -> first);

while(!q.empty()){

string curr = q.front();

q.pop();

for (const auto& neighbor : adjList[curr]){

if(visited.find(neighbor.first) == visited.end()){

visited.insert(neighbor.first);

q.push(neighbor.first);

}

}

}

return visited.size() == adjList.size(); //if all airports are visited a, graph is connected

}

};

int main(){

Graph flightGraph;

flightGraph.addFlight("New York","Los angeles",6);//assume 6 hours okay beta

flightGraph.addFlight("New York", "Chicago", 2);

flightGraph.addFlight("Los Angeles", "Chicago", 4);

flightGraph.addFlight("Los Angeles", "San Francisco", 3);

flightGraph.addFlight("Chicago", "San Francisco", 5);

if(flightGraph.isConnected()){

cout<< "The flight network is connected ";

}

else

cout << "The flight network is not connected ";

return 0;

}