SC165

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 takes 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 the airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Justify the storage representation used.

```
#include<iostream>
using namespace std;

class Graph {
   public:
   int noc;
   int nof;
   int start;
   int end;
   int cost,visited[30];
   int adj[10][10];

   Graph(){
      cout<<"Enter the no. of cities: ";
      cin>>noc;
      cout<<"Enter the no. of Flights: ";
      cin>>nof;

   }
      //cout<<"Enter the values in adjacency matrix: ";
      void create_graph() {
      int f;

      for (int i=0; i<noc; i++)   {
         visited[i]=0;
         for (int j=0; j<noc; j++){
            adj[i][j]=0;
         }
      }

      for(int i= 0; i<nof; i++)   {
         cout<<"Enter cities connected by flight"<<i+1<<": ";
         cin>>f;
         for(int j=0; j<f;j++)   {
```

```
            cout<<"Enter start city, end city and distance between two cities: ";
            cin>>start>>end>>cost;
            adj[start][end] = cost;
            adj[end][start] = cost;
        }
    }
    for(int i=0; i<noc; i++)    {
        for(int j=0;j<noc;j++){
            cout<<adj[i][j]<<"\t";
        }
        cout<<endl;
    }

}

void connected(){
    int i;
    bool flag = false;
    for(i=0;i<noc;i++){
        for(int j=0;j<noc;j++){
            if(adj[i][j]!=0 && visited[i]!=1){
                visited[i]=1;
            }
        }
    }
    for(i =0 ;i<noc;i++){
        if(visited[i]==0){
            break;
        }
    }
    if(i==noc){
        flag = true;
    }

    if(flag)
        cout<<"connected";
    else
        cout<<"not connected";
}
};

int main()  {
    Graph g;
    cout<<"Cities: "<<g.noc<<endl;
```

```
    cout<<"Flights: "<<g.nof<<endl;
    g.create_graph();
    g.connected();

}
```

Output

```
/tmp/C22r4QERe5.o
Enter the no. of cities: 4
Enter the no. of Flights: 2
Cities: 4
Flights: 2
Enter cities connected by flight1: 2
Enter start city, end city and distance between two cities: 0 1 200
Enter start city, end city and distance between two cities: 0 3 100
Enter cities connected by flight2: 3
Enter start city, end city and distance between two cities: 1 2 40
Enter start city, end city and distance between two cities: 2 3 20
Enter start city, end city and distance between two cities: 2 0 300
0   200 300 100
200 0   40  0
300 40  0   20
100 0   20  0
connected
```

Output

```
/tmp/C22r4QERe5.o
Enter the no. of cities: 4
Enter the no. of Flights: 2
Cities: 4
Flights: 2
Enter cities connected by flight1: 2
Enter start city, end city and distance between two cities: 0 1 200
Enter start city, end city and distance between two cities: 0 3 100
Enter cities connected by flight2: 0
0   200 0   100
200 0   0   0
0   0   0   0
100 0   0   0
not connected
```