**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. Justify the storage representation used**

#include<iostream>

#include<stdlib.h>

#include<string.h>

using namespace std;

struct node

{

string vertex;

int time;

node \*next;

};

class adjmat

{

int m[10][10],n,i,j;

char ch;

string v[20];

node \*head[20];

node \*temp=NULL;

public:

adjmat()

{

for(i=0;i<20;i++)

{

head[i]=NULL;

}

}

void getgraph();

void displaym();

};

void adjmat::getgraph()

{

cout<<"\n enter no. of cities(max. 20)";

cin>>n;

cout<<"\n enter name of cities";

for(i=0;i<n;i++)

cin>>v[i];

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

cout<<"\n If path is present between city "<<v[i]<<" and "<<v[j]<<"press y else n";

cin>>ch;

if(ch=='y')

{

cout<<"\n Enter time required to reach city "<<v[j]<<" from "<<v[i]<<" in minutes";

cin>>m[i][j];

}

else if(ch=='n')

{

m[i][j]=0;

}

else

{

cout<<"\n unknown entry";

}

}

}

}

void adjmat::displaym()

{

cout<<"\n";

for(j=0;j<n;j++)

{

cout<<"\t"<<v[j];

}

for(i=0;i<n;i++)

{

cout<<"\n "<<v[i];

for(j=0;j<n;j++)

{

cout<<"\t"<<m[i][j];

}

cout<<"\n";

}

}

int main()

{

int m;

adjmat a;

while(1)

{

cout<<"\n\n enter the choice";

cout<<"\n 1.enter graph";

cout<<"\n 2.display adjacency matrix for cities";

cout<<"\n 3.exit";

cin>>m;

switch(m)

{ case 1: a.getgraph();

break;

case 2: a.displaym();

break;

case 3: exit(0);

default: cout<<"\n unknown choice";

}

}

return 0;

}