**Aim:**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.

**PROGRAM :**

#include<iostream>

#include<stdlib.h>

#include<string.h>

using namespace std;

struct node

{ string vertex;

int time;

node \*next;

};

class adjmatlist

{ int m[10][10],n,i,j; char ch; string v[20]; node \*head[20]; node \*temp=NULL;

public:

adjmatlist()

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

{ head[i]=NULL; }

}

void getgraph();

void adjlist();

void displaym();

void displaya();

};

void adjmatlist::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]<<" then press enter y otherwise 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: "; }

}

}

adjlist();

}

void adjmatlist::adjlist()

{ cout<<"\n \*\*\*\*";

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

{ node \*p=new(struct node);

p->next=NULL;

p->vertex=v[i];

head[i]=p;

cout<<"\n"<<head[i]->vertex;

}

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

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

{

if(m[i][j]!=0)

{

node \*p=new(struct node);

p->vertex=v[j];

p->time=m[i][j];

p->next=NULL;

if(head[i]->next==NULL)

{ head[i]->next=p; }

else

{ temp=head[i];

while(temp->next!=NULL)

{ temp=temp->next; }

temp->next=p;

}

}

}

}

}

void adjmatlist::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";

}

}

void adjmatlist::displaya()

{

cout<<"\n adjacency list is: ";

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

{

if(head[i]==NULL)

{ cout<<"\n adjacency list not present: "; break; }

else

{

cout<<"\n"<<head[i]->vertex;

temp=head[i]->next;

while(temp!=NULL)

{ cout<<"-> "<<temp->vertex;

temp=temp->next; }

}

}

cout<<"\n path and time required to reach cities is: ";

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

{

if(head[i]==NULL)

{ cout<<"\n adjacency list not present"; break; }

else

{

temp=head[i]->next;

while(temp!=NULL)

{ cout<<"\n"<<head[i]->vertex;

cout<<"-> "<<temp->vertex<<"\n [time required: "<<temp->time<<" min ]";

temp=temp->next; }

}

}

}

int main()

{ int m;

adjmatlist 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.display adjacency list for cities";

cout<<"\n 4.exit";

cin>>m;

switch(m)

{ case 1: a.getgraph();

break;

case 2: a.displaym();

break;

case 3: a.displaya();

break;

case 4: exit(0);

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

}

}

}