                                 EXPERIMENT:-7﻿INPUT:-﻿#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++)﻿﻿         cin>>ch;﻿         if(ch=='y')﻿         { ﻿﻿           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;﻿﻿                            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";﻿         }﻿    }﻿    return 0;﻿}                            ﻿OUTPUT:-﻿          Enter the choice:﻿1. Enter graph﻿2. Display adjacency matrix for cities﻿3. Display adjacency list for cities﻿4. Exit﻿1﻿Enter number of cities (max. 20): 3﻿Enter names of cities: mumbai pune nashik﻿Is there a path between city mumbai and mumbai? (y/n): n﻿Is there a path between city mumbai and pune? (y/n): y﻿Enter time required to reach city pune from mumbai in minutes: 180﻿Is there a path between city mumbai and nashik? (y/n): y﻿Enter time required to reach city nashik from mumbai in minutes: 300﻿Is there a path between city pune and mumbai? (y/n): y﻿Enter time required to reach city mumbai from pune in minutes: 180﻿Is there a path between city pune and pune? (y/n): n﻿Is there a path between city pune and nashik? (y/n): y﻿Enter time required to reach city nashik from pune in minutes: 240﻿Is there a path between city nashik and mumbai? (y/n): y﻿Enter time required to reach city mumbai from nashik in minutes: 300﻿Is there a path between city nashik and pune? (y/n): y﻿Enter time required to reach city pune from nashik in minutes: 240﻿Is there a path between city nashik and nashik? (y/n): n﻿\*\*\*﻿mumbai﻿pune﻿nashik﻿Enter the choice:﻿1. Enter graph﻿2. Display adjacency matrix for cities﻿3. Display adjacency list for cities﻿4. Exit﻿2﻿                mumbai  pune    nashik﻿mumbai        0       180     300﻿pune              180     0       240﻿nashik.         300     240     0﻿Enter the choice:﻿1. Enter graph﻿2. Display adjacency matrix for cities﻿3. Display adjacency list for cities﻿4. Exit﻿3﻿Adjacency list is:﻿mumbai -> pune -> nashik﻿pune -> mumbai -> nashik﻿nashik -> mumbai -> pune﻿Path and time required to reach cities is:﻿mumbai -> pune﻿[Time required: 180 min]﻿mumbai -> nashik﻿[Time required: 300 min]﻿pune -> mumbai﻿[Time required: 180 min]﻿pune -> nashik﻿[Time required: 240 min]﻿nashik -> mumbai﻿[Time required: 300 min]﻿nashik -> pune﻿[Time required: 240 min]﻿Enter the choice:﻿1. Enter graph﻿2. Display adjacency matrix for cities﻿3. Display adjacency list for cities﻿4. Exit﻿4﻿--------------------------------﻿Process exited after 732.2 seconds with return value 0﻿Press any key to continue . . .﻿