Name: Om Chandrakant Bhavsar

Class: SE-A

Roll No: COSA75

Practical No. 6

```
#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 --------Menu--------";
   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";
   cout<<"\n Enter the choice: ";
   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:

```
 --------Menu--------
 1.Enter Graph
 2.Display adjacency matrix for cities
 3.Display adjacency list for cities
 4.Exit
 Enter the choice: 1

 enter no. of cities(max. 20): 3

enter name of cities: Pune
Nashik
Mumbai

 if path is present between city Pune and Pune then press enter y otherwise n: n

 if path is present between city Pune and Nashik then press enter y otherwise n: y

 enter time required to reach city Nashik from Pune in minutes: 120

 if path is present between city Pune and Mumbai then press enter y otherwise n: y

 enter time required to reach city Mumbai from Pune in minutes: 60

 if path is present between city Nashik and Pune then press enter y otherwise n: y

 enter time required to reach city Pune from Nashik in minutes: 120

 if path is present between city Nashik and Nashik then press enter y otherwise n: n

 if path is present between city Nashik and Mumbai then press enter y otherwise n: y

 enter time required to reach city Mumbai from Nashik in minutes: 140

 if path is present between city Mumbai and Pune then press enter y otherwise n: y

 enter time required to reach city Pune from Mumbai in minutes: 60

 if path is present between city Mumbai and Nashik then press enter y otherwise n: y

 enter time required to reach city Nashik from Mumbai in minutes: 140

 if path is present between city Mumbai and Mumbai then press enter y otherwise n: n

 ****
```

```
Pune
Nashik
Mumbai
 --------Menu--------
 1.Enter Graph
 2.Display adjacency matrix for cities
 3.Display adjacency list for cities
 4.Exit
 Enter the choice: 2

        Pune    Nashik  Mumbai
 Pune   0       120     60

 Nashik 120     0       140

 Mumbai 60      140     0

 --------Menu--------
 1.Enter Graph
 2.Display adjacency matrix for cities
 3.Display adjacency list for cities
 4.Exit
 Enter the choice: 3

 adjacency list is:
Pune-> Nashik-> Mumbai
Nashik-> Pune-> Mumbai
Mumbai-> Pune-> Nashik
 path and time required to reach cities is:
Pune-> Nashik
   [time required: 120 min ]
Pune-> Mumbai
   [time required: 60 min ]
Nashik-> Pune
   [time required: 120 min ]
Nashik-> Mumbai
   [time required: 140 min ]
Mumbai-> Pune
   [time required: 60 min ]
Mumbai-> Nashik
   [time required: 140 min ]
 --------Menu--------
 1.Enter Graph
 2.Display adjacency matrix for cities
 3.Display adjacency list for cities
 4.Exit
```