Experiment 6

Program Code:

```
#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<<"\nEnter 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<<"\nAdjacency list not present";  break;   }
                else
                {
                  cout<<"\n"<<head[i]->vertex;
                temp=head[i]->next;
                while(temp!=NULL)
                {  cout<<"-> "<<temp->vertex;
                  temp=temp->next;  }
                }
            }
          cout<<"\nPath and time required to reach cities is: ";
    for(i=0;i<n;i++)
    {
                if(head[i]==NULL)
                {   cout<<"\nAdjacency 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\n";
  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 graph:
2.Display adjacency matrix for cities:
3.Display adjacency list for cities:
4.Exit
1
Enter No. of cities(max. 20): 3
Enter name of cities: Mumbai Pune Nashik
If path is present between city Mumbai and Mumbai then press enter y otherwise n: n
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: 30
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: 20
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: 30
```

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: 40

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: 20

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: 40

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

****

Mumbai

Pune

Nashik

Enter the choice:

1.Enter graph:

2.Display adjacency matrix for cities:

3.Display adjacency list for cities:

4.Exit

2

	Mumbai	Pune	Nashik
Mumbai	0	30	20
Pune	30	0	40
Nashik	20	40	0

Enter the choice:

1.Enter graph:

2.Display adjacency matrix for cities:

3.Display adjacency list for cities:

4.Exit

3

Adjacency list is:

Mumbai-> Pune-> Nashik

Pune-> Mumbai-> Nashik

Nashik-> Mumbai-> Pune

Path and time required to reach cities is:

Mumbai-> Pune

[Time required: 30 Min]

Mumbai-> Nashik

[Time required: 20 Min]

Pune-> Mumbai

[Time required: 30 Min]

Pune-> Nashik

[Time required: 40 Min]

Nashik-> Mumbai

[Time required: 20 Min]

Nashik-> Pune

[Time required: 40 Min]

Enter the choice:

1.Enter graph:

2.Display adjacency matrix for cities:

3.Display adjacency list for cities:

4.Exit

4

Process exited after 128.5 seconds with return value 0

Press any key to continue . . .