ASSIGNMENT-8: GRAPH – SHORTEST PATH ALGORITHM

Code:

```
#include<iostream>
# define max 20
using namespace std;

char v1;
class graph
{
     int
g[max][max],n,c[max],ch[max],min_dist,client_dist,visit_dist;
     char
v[max],str[max][max],min_path[max],client_path[max],visit_path[m
ax];
public:
     graph(int m)
    {
          n=m;
          visit_dist=0;
          client_dist=0;
          min_dist=0;
          for(int i=0;i<n;i++)
          {
               for(int j=0;j<n;j++)
               {
                    g[i][j]=0;
               }
          }
    }

     void accept_v();
```

```
     void accept_e();
     void display();
     void dj_init(int);
     void dj_init();
     void dj(char,char,int);
     void dj_client(char,char);
};

void graph::accept_v()
{
     int i=0;
     cout<<"\n\t\t\tNames Of Landmarks";
     while(i<n)
     {
          cout<<"\nEnter Name of Landmark ["<<i+1<<"] : ";
          cin>>v[i];
          i++;
     }
     cout<<"\n";

}

void graph::accept_e()
{
     char ch;
     char v1,v2;
     int i,j,cst;

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

```
          {
               if(i==j)
                    {
                        g[i][j]=0;
                        continue;

                    }
               else
                    cout<<"\n\tDistance Between Landmark
["<<v[i]<<"]["<<v[j]<<"] :  ";
                cin>>cst;
                    g[i][j]=g[j][i]=cst;
          }

     cout<<"\n";

}

void graph::display()
{
     int i,j;
     i=0;

     cout<<"\n";
     while(i<n)
     {
          cout<<"\t"<<v[i];
          i++;
     }

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

```
	{
		cout<<"\n"<<v[i];
		for(j=0;j<n;j++)
		{
			cout<<"\t"<<g[i][j];
		}
	}
	cout<<"\n";
}

void graph::dj_init()
{
	int i,j;

	for(i=0;i<n;i++)
	{
		c[i]=9999;
		ch[i]=0;
		for(j=0;j<n;j++)
		{
			str[i][j]='-';
		}
	}
	cout<<"\n";
}


void graph::dj_init(int i )
{
	for(int j=0;j<n;j++)
```

```
     {
          str[i][j]='-';
     }
}


void graph::dj(char s,char d,int f)
{
     dj_init();
     int i,j,l,k,flag,min=999,cst=0;
     i=0;
     while(v[i]!=s)
     {
         i++;
     }

     c[i]=0;

     k=0;
     str[i][k]=v[i];
     do
     {
          ch[i]=1;
          min=999;
          for(j=0;j<n;j++)
          {
               flag=0;
               cst=c[i]+g[i][j];
               if(g[i][j]!=0 && i!=j && cst<c[j])
               {
```

```
                k=0;
                c[j]=cst;
                dj_init(j);
                while(flag==0)
                {
                     str[j][k]=str[i][k];
                     k++;
                     if(str[i][k]=='-')
                     {
                          flag=1;
                     }
                }
                str[j][k]=v[j];
           }
      }


      for(l=0;l<n;l++)
      {
           if(c[l]<=min&&ch[l]==0)
           {
                min=c[l];
                i=l;
           }
      }

}
while(v[i]!=d);

j=0;
```

```
      if(f==0)
      {
           min_dist=c[i];
      }

      else
      {
          client_dist=c[i];
      }

      if(f==0)
      {
           cout<<"Minimum Distance : "<<c[i]<<"\nShortest Path :
";
      }

      while(str[i][j]!='-')
      {
           if(f==0)
           {
                min_path[j]=str[i][j];
           }

           if(f==0)
           {
               cout<<" "<<str[i][j];
           }
           j++;
      }

      min_path[j]='\0';
```

```
     cout<<"\n";
     j=1;
     while(str[i][j]!='-'&&f==1)
     {
          client_path[j-1]=str[i][j];
          j++;
     }
     client_path[j-1]='\0';
     cout<<"\nShortest Distance From "<<v1<<" To All
Destinations:\n";
     for(int h=0;h<n;h++)
     {
          cout<<"\nFrom "<<v1<<" To "<<v[h]<<" Is:
"<<c[h]<<"\n";
          for(int m=0;m<n;m++)
          {
               cout<<" "<<str[h][m];
          }
     }

     cout<<"\n";
}

int main()
{
     int n,ch;
     char v2;
     cout<<"\nEnter Number of Landmarks : ";
     cin>>n;

     graph g(n);
```

```
     do
     {
          cout<<"\n1.Accept Names Of Landmarks.\n2.Accept
Distance Between Landmarks.\n3.Display Adjacency
Matrix.\n4.Display Shortest Distance.\n5.Exit.";
          cout<<"\nEnter Your Choice:  ";
          cin>>ch;

          switch(ch)
          {
          case 1:
               g.accept_v();
               break;
          case 2:
               g.accept_e();
               break;
          case 3:
               g.display();
               break;
          case 4:
               cout<<"\nEnter Source(Name Of Landmark From
Where You Want To Start) :  ";
               cin>>v1;
               cout<<"\nEnter Destination(Name Of Landmark
Where You Want To Reach) :  ";
               cin>>v2;
               g.dj(v1,v2,0);
               break;

          case 5:
              cout<<"You Have Successfully Exitted....";
```

```
            break;

         default:
            cout<<"INVALID CHOICE.";

        }

    }
    while(ch!=5);

    return 0;
}
```

OUTPUT:

```
Enter Number of Landmarks : 5

1.Accept Names Of Landmarks.
2.Accept Distance Between Landmarks.
3.Display Adjacency Matrix.
4.Display Shortest Distance.
5.Exit.
Enter Your Choice:  1

                        Names Of Landmarks
Enter Name of Landmark [1] : A

Enter Name of Landmark [2] : B

Enter Name of Landmark [3] : C
```

```
Enter Name of Landmark [4] : D

Enter Name of Landmark [5] : E


1.Accept Names Of Landmarks.
2.Accept Distance Between Landmarks.
3.Display Adjacency Matrix.
4.Display Shortest Distance.
5.Exit.
Enter Your Choice:  2

        Distance Between Landmark [A][B] :  2

        Distance Between Landmark [A][C] :  4

        Distance Between Landmark [A][D] :  1

        Distance Between Landmark [A][E] :  5

        Distance Between Landmark [B][C] :  6

        Distance Between Landmark [B][D] :  7

        Distance Between Landmark [B][E] :  3

        Distance Between Landmark [C][D] :  8

        Distance Between Landmark [C][E] :  9
```

```
        Distance Between Landmark [D][E] :  2


1.Accept Names Of Landmarks.
2.Accept Distance Between Landmarks.
3.Display Adjacency Matrix.
4.Display Shortest Distance.
5.Exit.
Enter Your Choice:  3

        A       B       C       D       E
A       0       2       4       1       5
B       2       0       6       7       3
C       4       6       0       8       9
D       1       7       8       0       2
E       5       3       9       2       0

1.Accept Names Of Landmarks.
2.Accept Distance Between Landmarks.
3.Display Adjacency Matrix.
4.Display Shortest Distance.
5.Exit.
Enter Your Choice:  4

Enter Source(Name Of Landmark From Where You Want To Start) :  A

Enter Destination(Name Of Landmark Where You Want To Reach) :  E

Minimum Distance : 3
Shortest Path :  A D E
```

```
Shortest Distance From A To All Destinations:

From A To A Is: 0
 A - - - -
From A To B Is: 2
 A B - - -
From A To C Is: 4
 A C - - -
From A To D Is: 1
 A D - - -
From A To E Is: 3
 A D E - -

1.Accept Names Of Landmarks.
2.Accept Distance Between Landmarks.
3.Display Adjacency Matrix.
4.Display Shortest Distance.
5.Exit.
Enter Your Choice:  5
You Have Successfully Exitted....
```