**Practical No: - 8. Write a program to find shortest path using single source shortest path.**

```
#include<iostream>
#include<conio.h>
using namespace std;
int i,j,n,k,m,v,p,totcost;
int cost[50][50],dist[50],S[50],path[50];
class ShortPath
{
public:
void getdata();
void shortest(int,int);
}obj;
void ShortPath::getdata()
{
int c;
cout<<"\n Enter number of Vertices:- ";
cin>>n;
cout<<"\n Enter the number of edges:- ";
cin>>m;
cout<<"\n Enter the values for EDGE and its COST: \n";
cout<<"\n Node1 Node2 Cost \n";
for(k=1;k<=m;k++)
{
cin>>i>>j>>c;
cost[i][j]=c;
}
for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
                if(cost[i][j]==0)
cost[i][j]=1000;
cout<<"\n Enter the initiaL VERTEX :- ";
cin>>v;
}
void ShortPath::shortest(int v,int n)
{
int min; for(i=1;i<=n;i++)
{ S[i]=0;
dist[i]=cost[v][i];
```

```
}
path[++p]=v;
S[v]=1;
dist[v]=0;
cout<<"\n Shortest Path from vertex"<<v<<"to all Destination is : \n ";
for(i=2;i<=n-1;i++)
{
k=-1;
min=1000;
for(j=1;j<=n;j++)
{
if(dist[j]<min && S[j]!=1)
{
min=dist[j];
k=j;
}
}
if(cost[v][k]<=dist[k])
        p=1;
path[++p]=k;
for(j=1;j<=p;j++)
        cout<<" "<<path[j];
cout<<endl;
S[k]=1;
for(j=1;j<=n;j++)
if(cost[k][j]!=1000 && dist[j]>=dist[k]+cost[k][j] && S[j]!=1)
        dist[j]=dist[k]+cost[k][j];
}
}
void main()
{
//clrscr();
obj.getdata();
obj.shortest(v,n);
getch();
}
```

**/*OUTPUT*/**

```
 Enter number of Vertices:- 6
```

Enter the number of edges:- 11

Enter the values for EDGE and its COST:

Node1 Node2 Cost
1 2 50
1 3 45
1 4 10
2 3 10
2 4 15
3 5 30
4 1 20
4 5 15
5 2 20
5 3 35
6 5 3

Enter the initiaL VERTEX :- 1

Shortest Path from vertex1to all Destination is :
1 4
1 4 5
1 4 5 2
1 3