Input:

```
#include<iostream>
#include<string.h>
using namespace std;
class Graph
{
char Vnames[10][10];
int cost[10][10],no;
public:
Graph();
void creat_graph();
void display();
int Position(char[]);
void kru();
void prims();
};
Graph::Graph()
{
no=0;
for(int i=0;i<10;i++)
for(int j=0;j<10;j++)
{
if(i==j)

cost[i][j]=0;

else

cost[i][j]=999;
```

```
}
}
void Graph::creat_graph()
{
char ans,Start[10],End[10];
int wt,i,j;
cout<<"Enter number of nodes:";
cin>>no;
cout<<"\n Enter vertex name:";
for(i=0;i<no;i++)
cin>>Vnames[i];
do
{
cout<<"\nEnter Start and end point of edge:";
cin>>Start>>End;
cout<<"Enter weight:";
cin>>wt;
i=Position(Start);
j=Position(End);
cost[j][i]=cost[i][j]=wt;
cout<<"\nMore Edges: ";
cin>>ans;
}while(ans=='y' || ans=='Y');
}
void Graph::display()
{
int i,j;
cout<<"\nAdjecancy Matrix\n\t";
for(i=0;i<no;i++)

cout<<"\t"<<Vnames[i];
```

```
for(i=0;i<no;i++)

{
cout<<"\n\t"<<Vnames[i];
for(j=0;j<no;j++)
cout<<"\t"<<cost[i][j];
}

}
int Graph::Position(char S[10])
{
int i;
for(i=0;i<10;i++)
if(strcmp(Vnames[i],S)==0)
break;
return i;
}
void Graph::kru()
{
int i,j,v[10],x,y,min_cost=0,min,gr=1,flag=0,temp,d;
for(i=0;i<no;i++)

v[i]=0;
cout<<"\n node1\tnode2\tweight";
while(flag==0)
{
min=999;
for(i=0;i<no;i++)

{ for(j=0;j<no;j++)
```

```
{ if(i!=j && cost[i][j]<min)
{
min=cost[i][j];
x=i;
y=j;
}
}
}
if(v[x]==0 && v[y]==0)
{ v[x]=gr;v[y]=gr; gr++; }
else if(v[x]==0 && v[y]!=0)

v[x]=v[y];
else if(v[y]==0 && v[x]!=0)
v[y]=v[x];
else if(v[y]!=0 && v[x]!=0)
{
d=v[y];
for(i=0;i<no;i++)
if(v[i]==d)
v[i]=v[x];
}
cost[x][y]=cost[y][x]=999;
cout<<"\n"<<Vnames[x]<<"\t"<<Vnames[y]<<"==>\t"<<min;
min_cost+=min;
temp=v[0]; flag=1;
for(i=0;i<no;i++)

{
if(temp!=v[i])
{ flag=0; break;}
```

```
}

}
cout<<"\nminimum path is of value "<<min_cost;
}
void Graph::prims()
{
int c=1,b,i,j,x,y,min_cost=0,min,v[10]={0};
char start[10]="\0";
cout<<"\nfrom which City you want to start:";
cin>>start;
b=Position(start);

v[b]=1;
cout<<"\n City1\tCity2\tDistance";
while(c<no)

{
min=999;
for(i=0;i<no;i++)
{
if(v[i])
{
for(j=0;j<no;j++)
{
if(cost[i][j]<min && v[j]==0)
{
min=cost[i][j];
x=i;y=j;
}
}
```

```
}
}

cout<<"\n"<< Vnames [x]<<"\t"<< Vnames [y]<<"\t"<<min;
min_cost+=min;
cost[x][y]=cost[y][x]=999;
v[y]=1;
c++;
}
cout<<"\nMinimum Total cost"<<min_cost;
}

main()
{
Graph G,G1;
G1.creat_graph();
G1.display();
G1.prims();
G.creat_graph();
G.display();
G.kru();
}
```

Output:

```
/tmp/Tb7Ge3xVIk.o
Enter number of nodes:4
Enter vertex name:A B C D
Enter Start and end point of edge:A B
Enter weight:12
More Edges: Y
Enter Start and end point of edge:B C
Enter weight:40
More Edges: Y
Enter Start and end point of edge:C D
Enter weight:9
More Edges: Y
Enter Start and end point of edge:A D
Enter weight:59
More Edges: N
Adjecancy Matrix
      A   B   C   D
    A 0   12  999 59
    B 12  0   40  999
    C 999 40  0   9
    D 59  999 9   0
from which City you want to start:B
City1   City2   Distance
B   A   12
B   C   40
C   D   9
Minimum Total cost61
```