```
/*Represnt a graph of your college campus using adjacency list/adjacency matrix.Nodes should represnt a various departments and link shold represntthe distance between them.Find minimum spanning tree using Krusakls alogrithm
*/
#include <iostream>
using namespace
std; class graph{
int g[20][20];
int e,v;
public:
void accept(); void
display(); void
dijkstra(int start);
}; void graph:: accept(){ int src, dest,
cost, i,j; cout<<"Enter the number of vertices: "; cin>>v;
cout<<"Enter the number of edges: ";
cin>>e; for(i=0; i<v; i++){
for(j=0; j<v;j++){
g[i][j]=0;
} } for(i=0; i<e; i++){
cout<<"\nEnter source vertex: ";
cin>>src;
cout<<"Enter destination vertex: ";
cin>>dest;
cout<<"Enter the cost of the edge: ";
cin>>cost; g[src][dest]=cost;
g[dest][src]=cost;
} } void
```

```
graph::display(){
int i,j; for(i=0;
i<v; i++){
cout<<endl;
for(j=0; j<v; j++){
cout<<g[i][j]<<"\t";
}
} }
void graph::dijkstra(int start){
int r[20][20],
visited[20],distance[20],from[20],i,j,cnt,mindst,next;
for(i=0; i<v; i++){ for(j=0; j<v; j++){
if(g[i][j]==0){ r[i][j]=999;
}
else{
r[i][j]=g[i][j];
}
} } for(i=0;
i<v; i++){ visited[i]=0;
from[i]=start;
distance[i]=r[start][i];
}
distance[start]=0;
visited[start]=1;
cnt=v; while(cnt>0){
mindst=999;
for(i=0; i<v; i++){
if((mindst>distance[i]) && visited[i]==0){
mindst=distance[i]; next=i;
```

```
} }
visited[next]=1; for(i=0; i<v; i++){
if(visited[i]==0 &&
distance[i]>(mindst+r[next][i])){
distance[i]=mindst+r[next][i];
from[i]=next;
}
} cnt--
; }
for(i=0; i<v; i++){
cout<<"\nDistance of "<<i<< " from "<<start<<" is "<<distance[i]<<endl<<"Path "<<i;
j=i; do{
j=from[j]; cout<<"<- "<<j;
}
while(j!=start);
} } int
main() {
graph g;
int s;
g.accept();
g.display();
cout<<"\nEnter the starting vertex: ";
cin>>s;
g.dijkstra(s); return
0;
}
```