```
// Kruskal's Algorithm

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define MAX 90
void sort();
void print();
void kruskal();
int find(int belongs[] ,int vertexno);
void unin(int belongs[], int c1 , int c2);

int graph[MAX][MAX],spanning[MAX][MAX];
int n;

typedef struct edge
{
    int u,v,w;
}edge;

typedef struct edgelist
{
    edge edgedata[30];
    int n;
}edgelist;

edgelist elist,spanlist;


void kruskal()
{
    int i,j,cno1,cno2,belongs[MAX];


    elist.n=0;
    for(i=0;i<n;i++)
    {
    for(j=0;j<n;j++)
    {
    if(graph[i][j]!=0)
    {
        elist.edgedata[elist.n].u=i;
        elist.edgedata[elist.n].v=j;
        elist.edgedata[elist.n].w=graph[i][j];
        elist.n++;
    }
    }
    }
    sort();
```

```
    for(i=0;i<n;i++)
    belongs[i]=i;

    spanlist.n=0;

    for(i=0;i<elist.n;i++)
    {
        cno1=find(belongs,elist.edgedata[i].u);
        cno2=find(belongs,elist.edgedata[i].v);
        if(cno1 != cno2) // if the edge does not cause a cycle
        {

spanlist.edgedata[spanlist.n]=elist.edgedata[i];spanlist.n+=1;
            unin(belongs,cno1,cno2);
        }
    }
    print();
}
int find(int belongs[] ,int vertexno)
{
    return(belongs[vertexno]);
}

void unin(int belongs[], int c1 , int c2)
{
    int i;
    for(i=0;i<n;i++)
    if(belongs[i]==c2) // merge two components
      belongs[i]=c1;
 }
void print()
{
    int i,cost=0;
    printf("The minimum cost spanning tree is\n");

    for(i=0;i<spanlist.n;i++)
    {
        printf("\n%d-%d    cost =
%d\n",spanlist.edgedata[i].u,spanlist.edgedata[i].v,spanlist.edged
ata[i].w);
        cost=cost+spanlist.edgedata[i].w;
    }
    printf("\nThe total cost of the minimum cost spanning tree is
%d\n",cost);
}
void sort()
{
    int i,j;
    edge temp;
```

```
    for(i=0;i<elist.n;i++)
    {
        for(j=0;j<elist.n;j++)
        {
            if(elist.edgedata[j].w>elist.edgedata[j+1].w)
            {
                temp=elist.edgedata[j];
                elist.edgedata[j]=elist.edgedata[j+1];
                elist.edgedata[j+1]=temp;

            }
        }
    }
}
int main()
{
    int i,j;
    printf("Enter the number of vertices\n");
    scanf("%d",&n);
    printf("Enter the graph\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&graph[i][j]);
        }
    }
    kruskal();
    return 0;
}
```