//Kruskal's algorithm for generating minimum spanning tree from weighted graph

#include<stdio.h>

#include<conio.h>

#define INF 1000

char vertex[10];

int wght[10][10];

int span\_wght[10][10];

int source;

struct Sort

{

int v1,v2;

int weight;

}que[20];

int n,ed,f,r;

int cycle(int s,int d)

{

int j,k;

if(source==d)

return 1;

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

if(span\_wght[d][j]!=INF && s!=j)

{

if(cycle(d,j))

return 1;

}

return 0;

}

void build\_tree()

{

int i,j,w,k,count=0;

for(count=0;count<n;f++)

{

i=que[f].v1;

j=que[f].v2;

w=que[f].weight;

span\_wght[i][j]=span\_wght[j][i]=w;

source=i;

k=cycle(i,j);

if(k)

span\_wght[i][j]=span\_wght[j][i]=INF;

else

count++;

}

}

void swap(int \*i,int \*j)

{

int t;

t=\*i;

\*i=\*j;

\*j=t;

}

main()

{

int i,j,k=0,temp;

int sum=0;

printf("\n\n\tKRUSKAL'S ALGORITHM TO FIND SPANNING TREE\n\n");

printf("\n\tEnter the No. of Nodes : ");

scanf("%d",&n);

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

{

printf("\n\tEnter %d value : ",i+1);

fflush(stdin);

scanf("%c",&vertex[i]);

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

{

wght[i][j]=INF;

span\_wght[i][j]=INF;

}

}

printf("\n\nGetting Weight\n");

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

for(j=i+1;j<n;j++)

{

printf("\nEnter 0 if path Doesn't exist

between %c to %c : ",vertex[i],vertex[j]);

scanf("%d",&ed);

if(ed>=1)

{

wght[i][j]=wght[j][i]=ed;

que[r].v1=i;

que[r].v2=j;

que[r].weight=wght[i][j];

if(r)

{

for(k=0;k<r;k++)

if(que[k].weight>que[r].weight)

{

swap(&que[k].weight,&que[r].weight);

swap(&que[k].v1,&que[r].v1);

swap(&que[k].v2,&que[r].v2);

}

}

r++;

}

}

printf("\n\tORIGINAL GRAPH WEIGHT MATRIX\n\n");

printf("\n\tweight matrix\n\n\t");

for(i=0;i<n;i++,printf("\n\t"))

for(j=0;j<n;j++,printf("\t"))

printf("%d",wght[i][j]);

build\_tree();

printf("\n\n\t\tMINIMUM SPANNING TREE\n\n");

printf("\n\t\tLIST OF EDGES\n\n");

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

for(j=i+1;j<n;j++)

if(span\_wght[i][j]!=INF)

{

printf("\n\t\t%c ------ %c = %d ",   
 vertex[i],vertex[j],span\_wght[i][j]);

sum+=span\_wght[i][j];

}

printf("\n\n\t\tTotal Weight : %d ",sum);

getch();

}