LAB TEST - 2

NAME: T. VAGHDEVI PRAVEEN
USN: 1BM19CS175
DATE: 5/07/2021
SIGNATURE: Vaghdevi

Find Minimum cost Spanning Tree of a given Undirected graph using Kruskals algorithm.

```
#include <stdio.h>
void main ()
{
int a[20][20], b[20][20], c[20][20], d[20][20],
 nod=0, n, val1 = 0, i, j, K, t, m=0, posx, posy, val;
printf ("\n Enter the value of n:");
scanf ("%d", &n);
printf ("\n Enter the adjacency matrix \n");
for (i=0; i<n; i++)
 {
 for (j=0; j<n; j++)
 {
 scanf ("%d", &a[i][j]);
 b[i][j] = (i==j ? 0 : a[i][j]);
 m=m+ (b[i][j] ? 1:0);
 c[i][j]=0;
 d[i] [j]=0;
 }
}
for (m=m/2; m!=0 && (nod!= (n-1)); m--)
 {
 val = 32767;
```

NAME: T.VAGHDEVI PRAVEEN
USN: 1BM19CS175
DATE: 5/07/2021
SIGNATURE: Vaghdevi

```
for (i=0;i<n;i++)
{
for (j=0;j<n;j++)
 {
  if (b[i][j]!=0&& b[i][j]<val)
  {
   posx=i;
   posy=j;
   val=b[i][j];
  }
 }
}
 b[posx][posy]=0;
 b[posy][posx]=0;
 if (c[posx][posy]==0).
  {
   c[posx][posy]=1;
   c[posy][posx]=1;
   for(k=0; k<n;k++)
    {
    for (i=0;i<n;i++)
     {
      for (j=0;j<n;j++)
     {
      c[i][j]=c[i][j]|(c[i][k]&c[k][j]);
     }
    }
   }
```

NAME: T.VAGHDEVI PRAVEEN
USN: 1BM19CS175
DATE: 5/7/2021
SIGNATURE: Vaghdevi

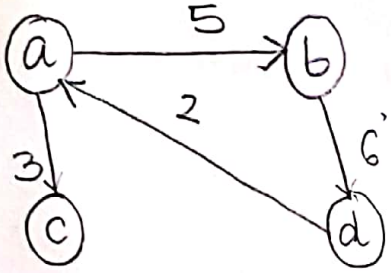
```
val1 = val1 + a[posx][posy];
nod = nod + 1;
d[posx][posy] = a[posx][posy];
d[posy][posx] = a[posy][posy];
}
}
if (nod==n-1)
{
for (i=0; i<n; i++)
{
printf("\n");
for (j=0; j<n; j++)
{
printf("%d", d[i][j]);
}
}
printf("\n Spanning tree has a cost of %d", val1);
}
else
{
printf("\n Spanning tree doesnot exist!");
}
}
```

NAME: T. VAGHDEVI PRAVEEN
USN: 1BM19CS175
DATE: 5/07/2021
SIGNATURE: Vaghdevi

Modification: While finding the MST using Kruskals algorithm, if you come accross a cycle print the vertices in the cycle.

```
if (nod && val 1)
{
printf("\n\n It is a connected, Acyclic graph!");
 }
if (!nod && val 1)
 {
 printf ("\n\n It is a not-connected, Acyclic graph!");
 }
 if (nod && !val 1)
 {
 printf ("\n\n Graph is a connected, Cyclic graph!");
 }
 if (!nod && !val 1)
 {
 printf ("\n\n It is not-connected, Cyclic graph!");
 }
 }
```

Graph :-



Adjacency Matrix:

	a	b	c	d
a	0	5	3	0
b	0	0	0	6
c	0	0	0	0
d	2	0	0	0

(4)