Name: N. Akhilesh
USN: 1BM19CS092

ADA

Lab-test-2

Sig
Section: 4B
Semester: 4th
Sign: Akhilesh

Find the Minimum cost spanning tree of a given undirected graph, using kruskals algorithm:-

```
#include<stdio.h>
#include<Process.h>
  void kruskals();
 int cost[10][10],n,i,j,sum,min;
 int count, k,u,v, Parent[10];
 int t[10][10];
 void union_ij(int,int);
 int find(int);
 int main()
{
  Printf("Enter the No: of vertices:");
  scanf("%d", &n);
  Printf("Enter the cost Adjacency matrix:\n");
  for(i=0; i<n;i++)
  {
   for(j=0; j<n;j++)
   {
  scanf("%d", &cost[i][j]);
  }
  kruskals();
  return 0;
}
void kruskals()
{
  count=0;
  k=0;
  sum=0;
  for(i=0;i<n;i++)
  Parent[i]=i;
```

```
while(count!=n-1)
{
  min=999;
  for(i=0;i<n;i++)
  {
    for(j=0;j<n;j++)
    {
      if(cost[i][j]<min && cost[i][j]!=0)
      {
        min=cost[i][j];
        u=i;
        v=j;
      }
    }
  }
  i=find(u);
  j=find(v);
  if(i!=j)
  {
    t[k][0]=u;
    t[k][1]=v;
    k++;
    count++;
    sum=sum+cost[u][v];
    union_ij(i,j);
  }
  cost[u][v]=cost[v][u]=999;
}
Printf("Minimum spanning tree:\n");
for(i=0;i<n-1;i++)
{
  Printf("%d->%d",t[i][0],t[i][1]);
}
Printf("Total cost=%d",sum);
}
void union_ij(int i,int j)
{
  if(i<j)
  Parent[j]=i;
  else
  Parent[i]=j;
}
int find(int v)
{
  while(Parent[v]!=v)
    v=Parent(v);
  return v;
}
```

Name: N. Akhilesh
USN : 1BM19CS092
Sign : Akhilesh

Modification :-

While finding the MST using Kruskal algorithm if you come across a cycle print the vertices in the cycle

```
void Kruskals()
{
  count=0;
  k=0;
  sum=0;
  for(i=0; i<n; i++)
  {
    parent[i]=i;
  }
  while(count!=(n-1))
  {
    min=999;
    for(i=0; i<n; i++)
    {
      for(j=0; j<n; j++)
      {
        if(cost[i][j]<min && cost[i][j]!=0)
        {
          min=cost[i][j];
          u=i;
          v=j;
        }
      }
    }
    i=find(u);
    j=find(v);
    if(i==j)
    {
      Printf("vertices forming cycle are: %d and
                %d\n", u, v);
    }
    if(i!=j)
    {
      t[k][0]=u;
      t[k][1]=v;
      k++;
      count++;
```

```
        sum = sum + cost[u][v];
        union_ij(i,j);
    }
    cost[u][v] = cost[v][u] = 999;
}
printf("Minimal spanning tree:\n");
for(i=0; i<k; i++)
{
    printf("%d -> %d", t[i][0], t[i][1]);
}
printf("\n Total cost = %d\n", sum);
}

int main()
{
printf("Enter the No: of vertices:");
scanf("%d", &n);
printf("Enter the cost Adjacency Matrix:\n");
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
    {
        scanf("%d", &cost[i][j]);
    }
}
kruskal();
return 0;
}
```