PRIMS ALGORITHM:

```
#include<stdio.h>
#include<conio.h>
int vis[10],vt[10],et[10][2],e=0,n;
float cost[10][10],sum=0;
void prims()
{
   int x=1,min,i,j,m,k,u,v;
   vt[x]=1;
   vis[x]=1;
   for(i=1;i<n;i++)
   {

      j=x;
      min=999;
      while(j>0)
      {
         k=vt[j];
         for(m=2;m<=n;m++)
         {
            if(cost[k][m]<min && vis[m]==0)
            {
               min=cost[k][m];
               u=k;
               v=m;
```

```
            }
          }
          j--;
        }
        vt[++x]=v;
        et[i][1]=u;
        et[i][2]=v;
        e++;
        vis[v]=1;
        sum=sum+cost[u][v];
    }
}
void main()
{
    int i,j;
    printf("Enter the number of vertices:");
    scanf("%d",&n);
    printf("\nEnter the cost matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%f",&cost[i][j]);
        }
    }
```

```
    prims();
    for(i=1;i<=e;i++)
    {
       printf("%d-->%d\n",et[i][1],et[i][2]);
    }
    printf("Total cost=%f",sum);
}
```

```
Enter the number of vertices:5

Enter the cost matrix:
0 1 5 2 999
1 0 999 999 999
3 999 0 3 999
2 999 3 0 1.5
999 999 999 1.5 0
1-->2
1-->4
4-->5
4-->3
Total cost=7.500000
```

DIJIKTRAS ALGORITHM:

```
#include<stdio.h>
#include<conio.h>
int vis[10],cost[10][10],dest[10];
int n,src;
void dijktras()
{
int i,count,min,u;
```

```
for(i=1;i<=n;i++)
{
  dest[i]=cost[src][i];
}
vis[src]=1;
count=1;
while(count<=n)
{
   min=999;
   for(i=1;i<=n;i++)
   {
      if(dest[i]<min && vis[i]==0)
      {
         min=dest[i];
         u=i;
      }
   }
   vis[u]=1;
   for(i=1;i<=n;i++)
   {
      if(dest[u]+cost[u][i]<dest[i] && vis[i]==0)
      {
         dest[i]=dest[u]+cost[u][i];
      }
      count++;
```

```
        printf("%d -> %d=%d\n",src,i,dest[i]);
    }
}


}
void main()
{
    int i,j;
    printf("enter the no of vetices:");
    scanf("%d",&n);
    printf("\nEnter the cost matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
        }
    }
    printf("\nenter the source vertex:");
    scanf("%d",&src);
    dijktras();
    getch();
}
```

```
enter the no of vetices:4

Enter the cost matrix:
0 3 999 7
3 0 4 2
999 4 0 5
7 2 5 0

enter the source vertex:1
1 -> 1=0
```

KRUSKALS ALGORITHM:

```
#include<stdio.h>

int cost[10][10],t[10][10],parent[10],n;

void kruskal()

{

 int i,j,u,v;

 int count=0;

 int k=1;

 int sum=0;

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

 {

   parent[i]=i;

 }

 while(count!=n-1)

 {

  int min=999;

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

```
 {
  for(j=1;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("Spanning Tree:\n");
```

```
 for(i=1;i<=count;i++)
 printf("%d->%d\t",t[i][0],t[i][1]);
 printf("\nTotal Cost=%d",sum);
 getch();
}

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;
```

```
}


int main()
{
 int i,j;
 printf("\nEnter the number of vertices:");
 scanf("%d",&n);
 printf("\Enter the cost matrix:");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
    scanf("%d",&cost[i][j]);
  }
 }
 kruskal();
 return 0;
}
```

```
Enter the number of vertices:6
←nter the cost matrix:0 3 999 999 6 2 3 0 1 999 999 4 999 1 0 6 999 4 999 999 6 0 8 5 6 999 999
Spanning Tree:
2->3    1->6    5->6    1->2    4->6
Total Cost=13
```