6.2 From a given vertex in a weighted Connected graph, Write a program to find Shortest paths to other vertices using Dijks-tra's algorithm. Draw simple, Connected weighted graph with 8 vertices and 16 edges, each with unique edge weight. Identify One vertex as a Start vertex and obtain Shortest path using Dijkstra's algorithm.

Soln)

```
#include <stdio.h>
#include <conio.h>
#define INFINITY 9999
#define MAX 10
Void dijkstra(int G[MAX][MAX], int n,
                        int startnode);

int main(){
    int G[MAX][MAX], i, j, n, u;
    printf("Enter no. of Vertices:");
    Scanf("%d", &n);
    printf("Enter the adjacency matrix:\n");
    for(i=0; i<n; i++)
    for(j=0; j<n; j++)
    scanf("%d", &G[i][j]);
```

```
printf("\n Enter the starting node : ");
scanf("%d", &u);
dijkstra(G, n, u);
return 0;
}

Void dijkstra(int G[MAX][MAX], int n,
              int startnode){

    int cost[MAX][MAX], distance[MAX], pred[MAX];
    int visited[MAX], count, mindistance,
        nextnode, i, j;

    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            if(G[i][j]==0)
                cost[i][j] = INFINITY;
            else
                cost[i][j] = G[i][j];

    for(i=0; i<n; i++){
        distance[i] = cost[startnode][i];
        pred[i] = startnode;
    } visited[i]=0;
```

```
distance [startnode ] = 0;
visited [startnode ] = 1;
count = 1;
while (count < n - 1) {
        mindistance = INFINITY;
for (i = 0; i < n; i++)
    if (distance [i] < mindistance &&
                        ! visited [i]) {
        mindistance = distance [i];
        nextnode = i;
    }
visited [nextnode ] = 1;
for (i = 0; i < n; i++)
    if (! visited [i])
    if (mindistance + cost [next node] [i]
                        < distance [i]) {
        distance [i] = mindistance +
                cost [next node ] [i];
        pred [i] = nextnode;
    }
count ++;
}
```

```
for (i=0; i<n; i++)
    if (i != startnode) {
        printf("\n Distance Of node %d = %d",
                                    i, distance[i]);
        printf("\n path = %d", i);
        j = i;
        do {
            j = pred [j];
            printf("<- %d ", j);
        } while (j != startnode);
    }
}
```

6.3) Write a program to find Minimum cost Spanning tree of a given undirect graph using Kruskal's algorithm.

Ans)
```
#include<stdio.h>
# define MAX 30
typedef struct edge{
        int u,v,w;
}edge;
type struct edgelist{
        edge data[MAX];
        int n;
}edgelist;
edgelist elist;
int G[MAX][MAX],n;
 edgelist spanlist;
void kruskal();
int find(int belongs[], int vertexno);
void union1(int belongs[], int c1, int c2);
void sort();
void print();
```

~~distance[Estart]~~

```
void main (){
    int i, j, total_cost;
    printf("\n Enter number of vertices:");
    scanf("%d",&n);
    printf("\n Enter the adjacency matrix:\n");
    for(i=0; i<n; i++)
      for(j=0;j<n;j++)
        scanf("%d", &G[i][j]);
        kruskal();
        print();
}

Void kruskal(){
  int belongs[MAX], i, j, cno1, cno2;
  elist.n=0;

  for(i=1; i<n; i++){
    for(j=0; j<i; j++){
      if(G[i][j]!=0){
        elist.data[elist.n].u=i;
        elist.data[elist.n].v=j;
        elist.data[elist.n].w=G[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.data[i].u);
        cno2=find(belongs, elist.data[i].v);
        if(cno1!=cno2){
            spanlist.data[spanlist.n]=
                                elist.data[i];
            spanlist.n = spanlist.n+1;
            union1(belongs, cno1, cno2);
        }
    }
}
int find(int belongs[], int vertexno){
    return(belongs[vertexno]);
}
void union1(int belongs[], int c1, int c2){
    int i;
    for(i=0; i<n; i++)
     if(belongs[i]==c2)
     belongs[i]=c1; }
```

```
Void Sort(){
    int i,j;
    edge temp;
    for(i=1;i<elist.n;i++)
     for(j=0;j<elist.n-1;j++)
if(elist.data[j].w>elist.data[j+1].w){
        temp = elist.data[j];
        elist.data[j] = elist.data[j+1];
        elist.data[j+1]=temp;
    }
}

Void print(){
    int i, cost=0;
    for(i=0; i<Spanlist.n;i++){
    printf("\n%d\t%d\t%d",
        Spanlist.data[i].u,
        Spanlist.data[i].v,
        Spanlist.data[i].w);
    Cost =cost+Spanlist.data[i].w;
    }
printf("\n\n Cost of the Spanning tree = %d"; Cost);
}
```