```
//Created By Ritwik Chandra Pandey
//On 5th Nov' 2021
//Dijkastras Shortest path algorithm : Given a graph G and source vertex S, Dijkstra's shortest path algorithm is used to find shortest paths from source S to all vertices in the given graph.


#include <limits.h>
#include <stdio.h>
#define MAX 20
int V,E;
int graph[MAX][MAX];
#define INFINITY 99999
void dijkstra(int G[MAX][MAX],int n,int startnode) {
	int cost[MAX][MAX];
	int distance[10],pred[10],visited[10],count,mindistance,nextnode,i,j;
	for(i=1;i<=n;i++){
		for(j=1;j<=n;j++){
			if(G[i][j]==0){
				cost[i][j]=INFINITY;
			}else{
				cost[i][j] = G[i][j];
			}
		}
	}
	for(i=1;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=1;i<=n;i++){
			if(distance[i]<mindistance && visited[i]==0){
				mindistance = distance[i];
				nextnode =i;
			}
```

```
                }
                visited[nextnode] = 1;
                for(i=1;i<=n;i++){
                        if(visited[i]==0){
                                if(mindistance+cost[nextnode][i] < distance[i]){
                                        distance[i] = mindistance + cost[nextnode][i];
                                        pred[i] = nextnode;
                                }
                        }
                }
                count++;
        }
        printf("Node\tDistance\tPath\n");
        for(i=1;i<=n;i++){
                if(i!=startnode){
                        if(distance[i]==INFINITY){
                                printf("%4d",i);
                                printf("\t");
                                printf("%8s","INF");
                                printf("\t");
                                printf("NO PATH");
                                printf("\n");
                        }else{
                                printf("%4d",i);
                                printf("\t");
                                printf("%8d\t",distance[i]);
                                printf("%d",i);
                                j=i;
                                do{
                                        j = pred[j];
                                        printf("<-");
                                        printf("%d",j);
                                }while(j!=startnode);
                                printf("\n");
                        }
                }
        }
}
int main() {
        int s,d,w,i,j;
```

```
	printf("Enter the number of vertices : ");
	scanf("%d",&V);
	printf("Enter the number of edges : ");
	scanf("%d",&E);
	for(i = 1 ; i <= V; i++) {
		for(j=1; j <= V; j++ ) {
			graph[i][i] = 0;
		}
	}
	for(i=1;i<=E;i++) {
		printf("Enter source : ");
		scanf("%d",&s);
		printf("Enter destination : ");
		scanf("%d",&d);
		printf("Enter weight : ");
		scanf("%d",&w);
		if(s > V || d > V || s<=0 || d<=0) {
			printf("Invalid index. Try again.\n");
			i--;
			continue;
		} else {
			graph[s][d] = w;
		}
	}
	printf("Enter the source :");
	scanf("%d",&s);
	dijkstra(graph, V,s);
	return 0;
}
```