PROGRAM 7

7)

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm. Write the program in Java.

```
package labprograms;

import java.util.Scanner;

class p7 {

        static void shortest(int v,int cost[][],int dist[],int n)

        {

                boolean[] s=new boolean[10];

                int i,w,u,num;

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

                {

                        s[i]=false;

                        dist[i]=cost[v][i];

                }

                s[v]=true;

                dist[v]=0;

                num=2;

                while(num<=n)

                {

                        u=choose(dist,s,n);

                        s[u]=true;

                        num++;

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

                        {

                                if(((dist[u]+cost[u][w])<dist[w])&&!s[w])

                                        dist[w]=dist[u]+cost[u][w];

                        }

                }

        }

        static int choose(int dist[],boolean s[],int n)
        {
                int w,j=1,min;
                min=9999;
                for(w=1;w<=n;w++)
                        if((dist[w]<min)&&(s[w]==false))
                        {
                                min=dist[w];
                                j=w;
                        }
                return j;
        }
        public static void main(String[] args) {
                int[][] cost=new int[50][50];
                int[] dist=new int[50];
                int i,j,n,v;
                Scanner sc=new Scanner(System.in);
                System.out.print("Enter the number of nodes : ");
                n=sc.nextInt();
                System.out.println("Enter the cost adjacency matrix,'1000' for no direct path : ");
                for(i=1;i<=n;i++)
                        for(j=1;j<=n;j++)
                                cost[i][j]=sc.nextInt();
                System.out.print("Enter the starting vertex : ");
                v=sc.nextInt();
                shortest(v,cost,dist,n);
                System.out.println("Shortest path from starting vertex and other vertices are : ");
                for(j=1;j<=n;j++)
                        System.out.println(v+"->"+j+"="+dist[j]);
                sc.close();
        }
}
```