Experiment No: 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.

```
import java.util.*;
public class Dijkstras {
 void dij(int sv,int  n,int cost[][],int dist[],int visited[],int path[])
       {
              int count=1,i,j,v=0,min,w;
              for(i=1;i<=n;i++)
              {
                     visited[i]=0;
                     dist[i]= cost[sv][i];
                     if(cost[sv][i]==999)
                           path[i]=0;
                     else
                           path[i]=sv;
              }
              visited[sv]=1;
              while(count<=n-1)
              {
              min=999;
              for(j=1;j<=n;j++)
              {
              if((dist[j]<min)&&(visited[j]==0))
                     {
                           min=dist[j];
                           v=j;
                     }
              }
              visited[v]=1;
              count++;
              for(w=1;w<=n;w++)
              {
                     if(dist[w]>(dist[v]+cost[v][w]))
                     {
                     dist[w]=dist[v]+cost[v][w];
                     path[w]=v;
                     }
              }
              }

       }
void print_dij(int sv,int n,int dist[],int visited[],int path[])
{
       int j,t;
       for(j=1;j<=n;j++)
       {
              if(visited[j]==1 &&j!=sv)
              {
                     System.out.print("Shortest Distance between");
                     System.out.println(sv +" ->"+ j+ "is " + dist[j]);
                     t=path[j];
                     System.out.print("the path is");
                     System.out.print("   " +j);
                     while(t!=sv)
                     {
                           System.out.print("<-" +t);
                           t= path[t];
                     }
```

```
                        System.out.println("<-" +sv);
                }
        }
}
        public static void main(String[] args) {
                int i,j,n,sv;
                int cost[][]= new int[10][10];
                int dist[]= new int[10],visited[]= new int[10];
                int path[]= new int[10];
                Scanner sc= new Scanner(System.in);
                System.out.println("Enter the number of vertices");
                n= sc.nextInt();
                System.out.println("enter  the cost matrix put 999 when there is no
edge");
                for(i=1;i<=n;i++)
                        for(j=1;j<=n;j++)
                                cost[i][j]= sc.nextInt();
                System.out.println("the cost matrix is");
                for(i=1;i<=n;i++)
                {
                        for(j=1;j<=n;j++)
                                System.out.print(cost[i][j]+"\t");
                        System.out.println();
                }
                System.out.println("enter the source vertex");
                sv= sc.nextInt();
                Dijkstras d= new Dijkstras ();
                d.dij(sv,n,cost,dist,visited,path);
                d.print_dij(sv,n,dist,visited,path);
                }

}


/*

Enter the number of vertices
4
enter  the cost matrix put 999 when there is no edge
0 8 5 9
999 0 999 999
999 4 0 1
999 1 999 0
the cost matrix is
0       8       5       9
999     0       999     999
999     4       0       1
999     1       999     0
enter the source vertex
1
Shortest Distance between1 ->2is 7
the path is  2<-4<-3<-1
Shortest Distance between1 ->3is 5
the path is  3<-1
Shortest Distance between1 ->4is 6
the path is  4<-3<-1 */
```