Program 8:- Write a program to find the shortest path between vertices using the bellman-ford algorithm

```java
import java.util.Scanner;
 public class ford
 {
   private int D[];
   private int num_ver;
   public static final int MAX_VALUE = 999;
    public ford(int num_ver)
    {
      this.num_ver = num_ver;
      D = new int[num_ver + 1];
    }

   public void BellmanFordEvaluation(int source, int A[][])
   {
    for (int node = 1; node <= num_ver; node++)
    {
       D[node] = MAX_VALUE;
    }

   D[source] = 0;

   for (int node = 1; node <= num_ver - 1; node++)
     {
       for (int sn = 1; sn <= num_ver; sn++)
       {
        for (int dn = 1; dn <= num_ver; dn++)
         {
          if (A[sn][dn] != MAX_VALUE)
          {
             if (D[dn] > D[sn]+ A[sn][dn])
                D[dn] = D[sn] + A[sn][dn];
          }
         }
        }
      }

   for (int sn = 1; sn <= num_ver; sn++)
   {
    for (int dn = 1; dn <= num_ver; dn++)
     {
     if (A[sn][dn] != MAX_VALUE)
     {
        if (D[dn] > D[sn]+ A[sn][dn])
System.out.println("The Graph contains negative egde cycle");                  }
    }
 }
}
```

```java
        for (int vertex = 1; vertex <= num_ver; vertex++)
        {
System.out.println("distance of source"+source+"to"+vertex+"is" + D[vertex]);
        }
    }

    public static void main(String[ ] args)
    {
        int num_ver = 0;
        int source;
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of vertices");
        num_ver = scanner.nextInt();

        int A[][] = new int[num_ver + 1][num_ver + 1];
        System.out.println("Enter the adjacency matrix");
        for (int sn = 1; sn <= num_ver; sn++)
        {
        for (int dn = 1; dn <= num_ver; dn++)
        {
            A[sn][dn] = scanner.nextInt();
            if (sn == dn)
            {
                A[sn][dn] = 0;
                continue;
            }
            if (A[sn][dn] == 0)
            {
                A[sn][dn] = MAX_VALUE;
            }
        }
        }


        System.out.println("Enter the source vertex");
        source = scanner.nextInt();
        ford b = new ford (num_ver);
        b.BellmanFordEvaluation(source, A);
        scanner.close();
        }
    }
```