```
import java.util.Scanner;
public class bellmanford
{  public int distance[];
public int numb_vert;
public static final int MAX_VALUE=999;
public bellmanford(int numb_vert)
{  this.numb_vert = numb_vert; distance = new
int[numb_vert+1];    }
public void BellmanfordpEvaluation(int source,int
adj_matrix[][])
{    for(int node=1;node<=numb_vert;node++)
distance[node]=MAX_VALUE;
distance[source]=0;
for(int node=1;node<=numb_vert-1;node++)
{  for(int src_node=1;src_node<=numb_vert;src_node++)
{  for(int dest_node=1;dest_node<=numb_vert;
dest_node++)
{  if(adj_matrix[src_node][dest_node]!=MAX_VALUE)
{  if(distance[dest_node] > distance[src_node] +
adj_matrix[src_node][dest_node])
distance[dest_node] = distance[src_node] +
adj_matrix[src_node][dest_node];  }  }  } }
for(int src_node=1;src_node<=numb_vert;src_node++)
{  for(int dest_node=1;dest_node<=numb_vert;
dest_node++)
{  if(adj_matrix[src_node][dest_node]!=MAX_VALUE)
{  if(distance[dest_node] > distance[src_node] +
adj_matrix[src_node][dest_node])
{  System.out.println("The graph contains negative edge
cycle");    }  }   }   }
System.out.println("Routing Table for Router " + source+"
is");
System.out.println("Destination Distance\t");
```

```
for(int vertex=1;vertex<=numb_vert;vertex++)
System.out.println(+vertex+"\t\t\t"+distance[vertex]);   }
public static void main(String args[])
{  int numb_vert=0;
int source;
Scanner scan = new Scanner(System.in);
System.out.println("Enter the number of vertices");
numb_vert = scan.nextInt();
int adj_matrix[][] = new int[numb_vert+1][numb_vert+1];
System.out.println("Enter the adjacency matrix");
for(int src_node=1;src_node<=numb_vert;src_node++)
for(int dest_node=1;dest_node<=numb_vert;dest_node++)
{  adj_matrix[src_node][dest_node] = scan.nextInt();
if(src_node==dest_node)
{  adj_matrix[src_node][dest_node]=0;
continue;   }
if(adj_matrix[src_node][dest_node]==0)
adj_matrix[src_node][dest_node]=MAX_VALUE;   }
for(int i=1;i<=numb_vert;i++)
{  bellmanford bellmanford = new
bellmanford(numb_vert);
bellmanford.BellmanfordpEvaluation(i,adj_matrix);
}
scan.close();   }   }
```

Output 1 –

```
Enter the number of vertices 6
Enter the adjacency matrix
0   2   5   1   999 999
2   0   3   2   999 999
5   3   0   3   1   5
1   2   3   0   1   999
999 999 1   1   0   2
999 999 5   999 2   0
```