Date: 17/09/2025

Lab Practical #13:

To develop network using distance vector routing protocol and link state routing protocol.

Practical Assignment #13:

1. **C/Java Program: Distance Vector Routing Algorithm using Bellman Ford's Algorithm.**

```java
import java.util.*;

public class DistanceVectorRouting {
    private static final int INF = 9999;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of routers: ");
        int numRouters = scanner.nextInt();

        int[][] costMatrix = new int[numRouters][numRouters];
        System.out.println("Enter the cost matrix (use " + INF + " for infinity):");
        for (int i = 0; i < numRouters; i++) {
            for (int j = 0; j < numRouters; j++) {
                costMatrix[i][j] = scanner.nextInt();
            }
        }

        int[][] distanceVector = new int[numRouters][numRouters];
        int[][] nextHop = new int[numRouters][numRouters];

        for (int i = 0; i < numRouters; i++) {
            for (int j = 0; j < numRouters; j++) {
                distanceVector[i][j] = costMatrix[i][j];
                nextHop[i][j] = (costMatrix[i][j] != INF && i != j) ? j : -1;
            }
        }

        boolean updated;
        do {
            updated = false;
            for (int i = 0; i < numRouters; i++) {
```

Date: 17/09/2025

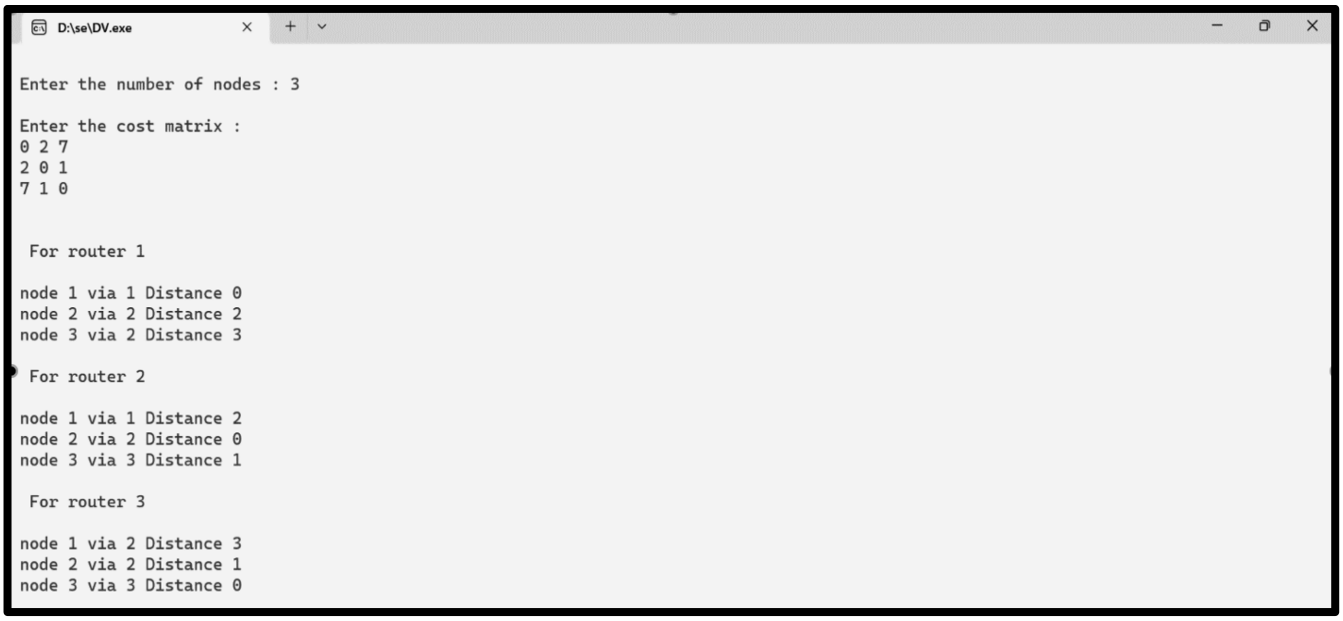
```
                for (int j = 0; j < numRouters; j++) {
                    for (int k = 0; k < numRouters; k++) {
                        if (distanceVector[i][k] + distanceVector[k][j] <
distanceVector[i][j]) {
                            distanceVector[i][j] = distanceVector[i][k] +
distanceVector[k][j];
                            nextHop[i][j] = nextHop[i][k];
                            updated = true;
                        }
                    }
                }
            }
        } while (updated);

        System.out.println("\nFinal Distance Vector Table:");
        for (int i = 0; i < numRouters; i++) {
            System.out.println("Router " + (i + 1) + ":");
            for (int j = 0; j < numRouters; j++) {
                if (distanceVector[i][j] == INF) {
                    System.out.print("INF ");
                } else {
                    System.out.print((distanceVector[i][j] + 1) + " ");
                }
            }
            System.out.println();
        }

        scanner.close();
    }
}
```

Date: 17/09/2025

Output:


D:\se\DV.exe

```
Enter the number of nodes : 3

Enter the cost matrix :
0 2 7
2 0 1
7 1 0


 For router 1

node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 2 Distance 3

 For router 2

node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 1

 For router 3

node 1 via 2 Distance 3
node 2 via 2 Distance 1
node 3 via 3 Distance 0
```

2. C/Java Program: Link state routing algorithm.

```
import java.util.*;

public class LinkStateRouting {
    private static final int INF = 9999;

    private static int minDistance(int[] dist, boolean[] visited, int n) {
        int min = INF, minIndex = -1;
        for (int v = 0; v < n; v++) {
            if (!visited[v] && dist[v] < min) {
                min = dist[v];
                minIndex = v;
            }
        }
        return minIndex;
    }

    private static void printPath(int[] prev, int j) {
        if (prev[j] == -1) {
            System.out.print((j + 1));
            return;
        }
```

Date: 17/09/2025

```
        printPath(prev, prev[j]);
        System.out.print(" -> " + (j + 1));
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of routers: ");
        int numRouters = scanner.nextInt();

        int[][] costMatrix = new int[numRouters][numRouters];
        System.out.println("Enter the cost matrix (use " + INF + " for infinity):");
        for (int i = 0; i < numRouters; i++) {
            for (int j = 0; j < numRouters; j++) {
                costMatrix[i][j] = scanner.nextInt();
            }
        }

        for (int src = 0; src < numRouters; src++) {
            int[] dist = new int[numRouters];
            int[] prev = new int[numRouters];
            boolean[] visited = new boolean[numRouters];

            Arrays.fill(dist, INF);
            Arrays.fill(prev, -1);

            dist[src] = 0;

            for (int count = 0; count < numRouters - 1; count++) {
                int u = minDistance(dist, visited, numRouters);
                if (u == -1) break;
                visited[u] = true;

                for (int v = 0; v < numRouters; v++) {
                    if (!visited[v] && costMatrix[u][v] != INF && dist[u] != INF &&
                        dist[u] + costMatrix[u][v] < dist[v]) {
                        dist[v] = dist[u] + costMatrix[u][v];
                        prev[v] = u;
                    }
                }
```

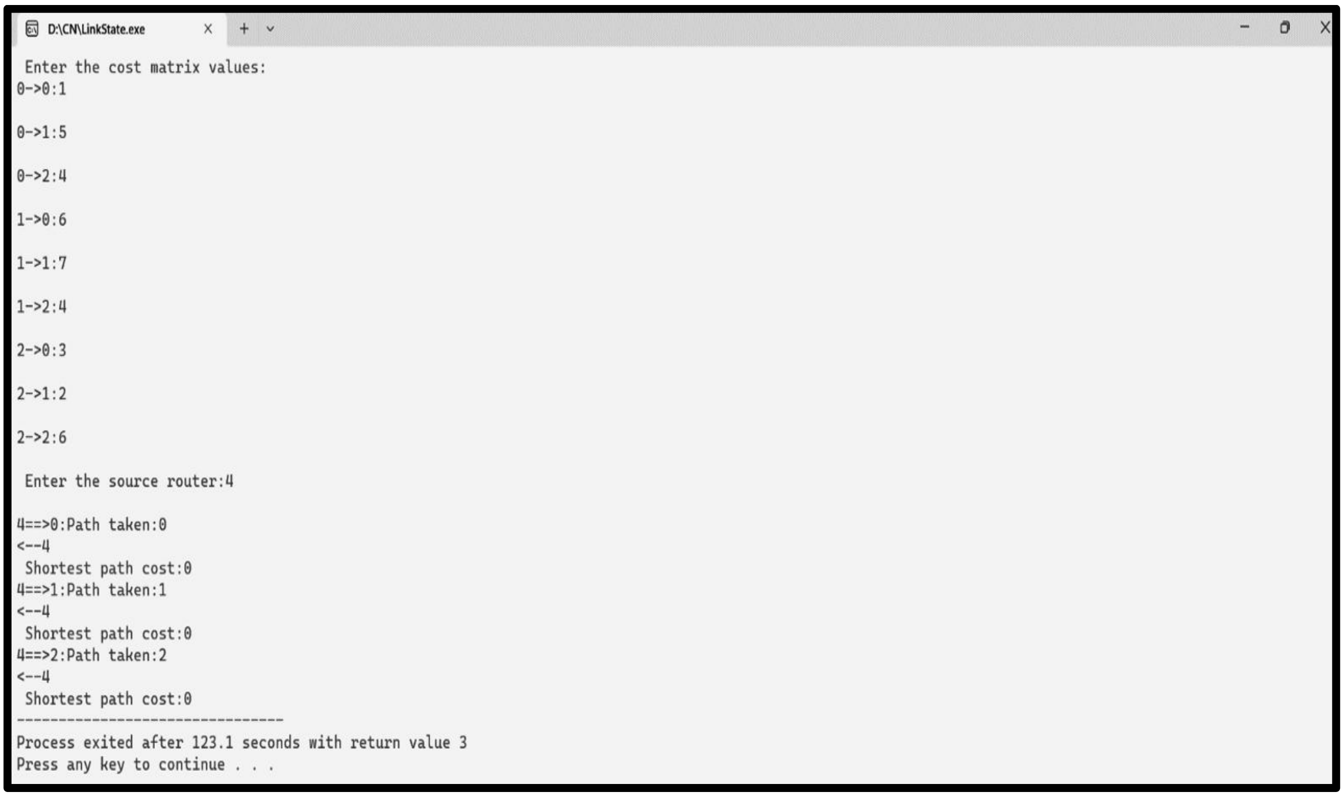
Date: 17/09/2025

```
            }

            System.out.println("\nRouting Table for Router " + (src + 1) + ":");
            System.out.println("Destination\tCost\tPath");
            for (int dest = 0; dest < numRouters; dest++) {
                if (dist[dest] == INF) {
                    System.out.println((dest + 1) + "\t\tINF\tNo path");
                } else {
                    System.out.print((dest + 1) + "\t\t" + dist[dest] + "\t");
                    printPath(prev, dest);
                    System.out.println();
                }
            }
        }

        scanner.close();
    }
}
```

Output:



```
D:\CN\LinkState.exe

 Enter the cost matrix values:
0->0:1

0->1:5

0->2:4

1->0:6

1->1:7

1->2:4

2->0:3

2->1:2

2->2:6

 Enter the source router:4

4==>0:Path taken:0
<--4
 Shortest path cost:0
4==>1:Path taken:1
<--4
 Shortest path cost:0
4==>2:Path taken:2
<--4
 Shortest path cost:0
--------------------------------
Process exited after 123.1 seconds with return value 3
Press any key to continue . . .
```