Date: 11/09/2025

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.**

```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

// Define maximum number of vertices
#define MAX_VERTICES 1000
// Define infinity for initialization
#define INF INT_MAX

// Define Edge structure
typedef struct {
   int source;
   int destination;
   int weight;
} Edge;

// Define Bellman-Ford function
void bellmanFord(int graph[MAX_VERTICES][MAX_VERTICES],
                 int vertices, int edges, int source)
{
   // Declare distance array
   int distance[MAX_VERTICES];

   // Initialize distances from source to all vertices as
   // infinity
   for (int i = 0; i < vertices; ++i)
      distance[i] = INF;
   // Distance from source to itself is 0
   distance[source] = 0;

   // Relax edges V-1 times
```

Date: 11/09/2025

```
    for (int i = 0; i < vertices - 1; ++i) {
        // For each edge
        for (int j = 0; j < edges; ++j) {
            // If the edge exists and the new distance is
            // shorter
            if (graph[j][0] != -1
                && distance[graph[j][0]] != INF
                && distance[graph[j][1]]
                       > distance[graph[j][0]]
                             + graph[j][2])
                // Update the distance
                distance[graph[j][1]]
                    = distance[graph[j][0]] + graph[j][2];
        }
    }

    // Check for negative cycles
    for (int i = 0; i < edges; ++i) {
        // If a shorter path is found, there is a negative
        // cycle
        if (graph[i][0] != -1
            && distance[graph[i][0]] != INF
            && distance[graph[i][1]]
                   > distance[graph[i][0]] + graph[i][2]) {
            printf("Negative cycle detected\n");
            return;
        }
    }

    // Print shortest distances from source to all vertices
    printf("Vertex   Distance from Source\n");
    for (int i = 0; i < vertices; ++i)
        printf("%d \t\t %d\n", i, distance[i]);
}

// Define main function
int main()
{
```

Date: 11/09/2025

```
    // Define number of vertices and edges
    int vertices = 6;
    int edges = 8;

    // Define graph as an array of edges
    int graph[MAX_VERTICES][MAX_VERTICES]
      = { { 0, 1, 5 }, { 0, 2, 7 }, { 1, 2, 3 },
          { 1, 3, 4 }, { 1, 4, 6 }, { 3, 4, -1 },
          { 3, 5, 2 }, { 4, 5, -3 } };

    // Call Bellman-Ford function with source vertex as 0
    bellmanFord(graph, vertices, edges, 0);
    return 0;
}
```

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

```
#include <stdio.h>

#define INF 999
#define MAX 6

int main() {
    // Number of nodes
    int n = 6;
    // Source node
    int src = 0;

    // Static cost matrix
    int cost[MAX][MAX] = {
        {0,   4,   6, INF, INF, INF},
        {4,   0,   2,   5, INF, INF},
        {6,   2,   0,   3,   4, INF},
        {INF, 5,   3,   0,   2,   3},
        {INF, INF, 4,   2,   0,   5},
        {INF, INF, INF, 3,   5,   0}
    };
```

Date: 11/09/2025

```
    int dist[MAX], visited[MAX];

    // Initialize
    for (int i = 0; i < n; i++) {
        dist[i] = cost[src][i];
        visited[i] = 0;
    }
    dist[src] = 0;
    visited[src] = 1;
    // Dijkstra
    for (int count = 1; count < n - 1; count++) {
        int min = INF, u = -1;

        // Find unvisited node with smallest distance
        for (int i = 0; i < n; i++) {
            if (!visited[i] && dist[i] < min) {
                min = dist[i];
                u = i;
            }
        }
        visited[u] = 1;

        // Update distances
        for (int v = 0; v < n; v++) {
            if (!visited[v] && dist[v] > dist[u] + cost[u][v]) {
                dist[v] = dist[u] + cost[u][v];
            }
        }
    }

    // Print results
    printf("Shortest distances from node %d:\n", src);
    for (int i = 0; i < n; i++) {
        printf("To node %d = %d\n", i, dist[i]);
    }

    return 0;
}
```