Exp_15\a_Floyd.c

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

#define V 4
#define INF 99999

void printSolution(int dist[][V]);

void floydWarshall(int graph[][V]) {
    int dist[V][V], i, j, k;

    // Initialize the solution matrix same as input graph matrix
    for (i = 0; i < V; i++) {
        for (j = 0; j < V; j++) {
            dist[i][j] = graph[i][j];
        }
    }

    // Add all vertices one by one to the set of intermediate vertices
    for (k = 0; k < V; k++) {
        // Pick all vertices as source one by one
        for (i = 0; i < V; i++) {
            // Pick all vertices as destination for the above picked source
            for (j = 0; j < V; j++) {
                // If vertex k is on the shortest path from i to j,
                // then update the value of dist[i][j]
                if (dist[i][k] != INF && dist[k][j] != INF &&
                    dist[i][k] + dist[k][j] < dist[i][j]) {
                    dist[i][j] = dist[i][k] + dist[k][j];
                }
            }
        }
    }

    printSolution(dist);
}

void printSolution(int dist[][V]) {
    printf("Following matrix shows the shortest distances between every pair of vertices:\n");
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            if (dist[i][j] == INF)
                printf("%7s", "INF");
            else
                printf("%7d", dist[i][j]);
        }
        printf("\n");
    }
}

int main() {
```

```
    int graph[V][V] = {
        {0, 5, INF, 10},
        {INF, 0, 3, INF},
        {INF, INF, 0, 1},
        {INF, INF, INF, 0}
    };

    floydWarshall(graph);

    return 0;
}
```