5 a) Design and implement C/C++ Program to solve All-Pairs Shortest Paths problem using Floyd's algorithm

#include <stdio.h>

// Function to print the solution matrix

void printSolution(int dist[][10], int V) {

printf("The 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] == 999)

printf("INF\t");

else

printf("%d\t", dist[i][j]);

}

printf("\n");

}

}

// Function to solve All-Pairs Shortest Paths problem using Floyd's algorithm

void floydWarshall(int graph[][10], int V) {

int dist[10][10];

int i, j, k;

// Initialize the solution matrix same as the input graph matrix

for (i = 0; i < V; i++)

for (j = 0; j < V; j++)

dist[i][j] = graph[i][j];

// Update dist[][] using all vertices as 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] + dist[k][j] < dist[i][j])

dist[i][j] = dist[i][k] + dist[k][j];

}

}

}

// Print the shortest distance matrix

printSolution(dist, V);

}

int main() {

int V;

printf("Enter the number of vertices: ");

scanf("%d", &V);

int graph[10][10];

printf("Enter the adjacency matrix:\n");

for (int i = 0; i < V; i++) {

for (int j = 0; j < V; j++) {

scanf("%d", &graph[i][j]);

}

}

// Solve All-Pairs Shortest Paths problem using Floyd's algorithm

floydWarshall(graph, V);

return 0;

}

OUTPUT:

Enter the number of vertices: 4

Enter the adjacency matrix:

0 5 999 10

999 0 3 999

999 999 0 1

999 999 999 0

The following matrix shows the shortest distances between every pair of vertices:

0 5 8 9

INF 0 3 4

INF INF 0 1

INF INF INF 0