15.FLOYD TECHNIQUE

''#include <stdio.h>

#define INF 99999

void floydWarshall(int graph[][100], int V)

{

int dist[V][V];

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

{

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

{

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

}

}

for (int k = 0; k < V; k++)

{

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

{

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

{

if (dist[i][k] + dist[k][j] < dist[i][j])

{

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

}

}

}

}

printf("Shortest distance matrix:\n");

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

{

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

{

if (dist[i][j] == INF)

printf("INF\t");

else

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

}

printf("\n");

}

}

int main()

{

int V;

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

while (scanf("%d", &V) != 1 || V <= 0)

{

printf("Invalid input. Please enter a positive integer: ");

while (getchar() != '\n');

}

int graph[100][100];

printf("Enter the adjacency matrix (use 'INF' for unreachable edges):\n");

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

{

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

{

int input;

scanf("%d", &input);

while (input < -1)

{

printf("Invalid input. Please enter a valid value (>= -1): ");

while (getchar() != '\n');

scanf("%d", &input);

}

if (input == -1)

{

graph[i][j] = INF;

}

else

{

graph[i][j] = input;

}

}

}

floydWarshall(graph, V);

return 0;

}

OUTPUT

