**TERMWORK 7**

#include <stdio.h>

#include <stdlib.h>

#define INF 999

int min(int a,int b);

void floyds(int p[10][10], int n) {

int i, j, k;

for (k = 1; k <= n; k++) {

for (i = 1; i <= n; i++) {

for (j = 1; j <= n; j++) {

if (i == j)

p[i][j] = 0;

else

p[i][j] = min(p[i][j], p[i][k] + p[k][j]);

}

}

}

}

int min(int a, int b) {

if (a < b)

return a;

else

return b;

}

int main(int argc, char \*argv[]) {

int p[10][10], n, e, u, v, w;

int i, j;

printf("\nEnter the number of vertices: ");

scanf("%d", &n);

printf("\nEnter the number of edges: ");

scanf("%d", &e);

// Initialize distance matrix with INF (999)

for (i = 1; i <= n; i++) {

for (j = 1; j <= n; j++) {

if (i == j)

p[i][j] = 0; // Initialize diagonal elements to 0

else

p[i][j] = INF;

}

}

// Input edges and weights

for (i = 1; i <= e; i++) {

printf("\nEnter the end vertices of edge %d with its weight: ", i);

scanf("%d %d %d", &u, &v, &w);

p[u][v] = w;

}

// Output the matrix of input data

printf("\nMatrix of input data:\n");

for (i = 1; i <= n; i++) {

for (j = 1; j <= n; j++) {

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

printf("999\t");

else

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

}

printf("\n");

}

// Apply Floyd-Warshall algorithm

floyds(p, n);

// Output the shortest paths

printf("\nThe shortest paths are:\n");

for (i = 1; i <= n; i++) {

for (j = 1; j <= n; j++) {

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

printf("999\t");

else if (i == j)

printf("0\t"); // Print 'o' for diagonal elements

else

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

}

printf("\n");

}

return 0;

}