//All Pair Shortest Path

//Floyd Warshal

//Complexity : O(V^3)

//Use if V <= 400

#define MAX 100

#define INF 1e7

int G[MAX][MAX], parent[MAX][MAX];

void graphINIT() {

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

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

G[i][j] = INF;

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

G[i][i] = 0;

}

void floydWarshall(int V){

//path printing matrix initialization

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

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

parent[i][j] = i; //we can go to j from i by only obtaining i (by default)

for(int k = 0; k < V; k++) //Selecting a middle point as k

for(int i = 0; i < V; i++) //Selecting all combination of source (i) and destination (j)

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

if(G[i][k] != INF && G[k][j] != INF) {

//if the graph contains negative edges, then min(INF, INF+ negative edge) = +-INF!

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

//if G[i][i] = negative, then node i is in negative circle

parent[i][j] = parent[k][j]; //if path printing needed

}

}

void printPath(int i, int j) {

if(i != j) printPath(i, parent[i][j]);

printf(" %d", j);

}

void minMax(int V) {

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

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

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

G[i][j] = min(G[i][j], max(G[i][k], G[k][j]));

}

void transitiveClosure(int V) {

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

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

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

G[i][j] |= (G[i][k] & G[k][j]);

}