#include <stdio.h>

#define INFINITY 9999 // Define a large value for unreachable nodes

struct node {

unsigned dist[20];

unsigned from[20];

} rt[10];

int main() {

int costmat[20][20];

int nodes, i, j, k, count = 0;

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

scanf("%d", &nodes);

printf("\nEnter the cost matrix (Enter 0 for same node, large value for no direct connection):\n");

for (i = 0; i < nodes; i++) {

for (j = 0; j < nodes; j++) {

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

if (i == j) {

costmat[i][j] = 0; // Ensure self-distance is always 0

} else if (costmat[i][j] == 0) {

costmat[i][j] = INFINITY; // Mark unreachable nodes

}

rt[i].dist[j] = costmat[i][j]; // Initialize the distance equal to the cost matrix

rt[i].from[j] = j; // Initialize the source node

}

}

// Distance vector algorithm loop

do {

count = 0;

for (i = 0; i < nodes; i++) {

for (j = 0; j < nodes; j++) {

for (k = 0; k < nodes; k++) {

// Check if the path through node k is shorter

if (rt[i].dist[j] > costmat[i][k] + rt[k].dist[j]) {

rt[i].dist[j] = costmat[i][k] + rt[k].dist[j];

rt[i].from[j] = k;

count++; // Track changes to determine when to stop

}

}

}

}

} while (count != 0); // Repeat until no updates are made

// Print routing tables

for (i = 0; i < nodes; i++) {

printf("\n\nRouting table for router %d\n", i + 1);

for (j = 0; j < nodes; j++) {

printf("Node %d via %d Distance: ", j + 1, rt[i].from[j] + 1);

if (rt[i].dist[j] == INFINITY)

printf("∞ (Unreachable)\n");

else

printf("%d\n", rt[i].dist[j]);

}

}

printf("\n");

return 0;

}