#include <stdio.h>

#include <limits.h>

#define MAX\_NODES 100

// Function to perform Bellman-Ford algorithm

void bellmanFord(int graph[MAX\_NODES][MAX\_NODES], int numNodes, int source) {

int distance[MAX\_NODES];

// Initialize distances from source to all other vertices as INFINITY

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

distance[i] = INT\_MAX;

}

distance[source] = 0;

// Relax all edges |V| - 1 times

for (int i = 0; i < numNodes - 1; ++i) {

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

for (int k = 0; k < numNodes; ++k) {

if (graph[j][k] != 0 && distance[j] != INT\_MAX && distance[j] + graph[j][k] < distance[k]) {

distance[k] = distance[j] + graph[j][k];

}

}

}

}

// Check for negative-weight cycles

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

for (int k = 0; k < numNodes; ++k) {

if (graph[j][k] != 0 && distance[j] != INT\_MAX && distance[j] + graph[j][k] < distance[k]) {

printf("Graph contains negative weight cycle\n");

return;

}

}

}

// Print the distance array

printf("Vertex Distance from Source\n");

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

printf("%d \t\t %d\n", i, distance[i]);

}

}

// Main function

int main() {

int numNodes;

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

scanf("%d", &numNodes);

int graph[MAX\_NODES][MAX\_NODES];

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

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

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

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

}

}

int source;

printf("Enter the source vertex: ");

scanf("%d", &source);

bellmanFord(graph, numNodes, source);

return 0;

}