#include <stdio.h>

#define MAX 100

#define INF 99999

int graph[MAX][MAX]; // Adjacency matrix

int dist[MAX]; // Distance from source

int visited[MAX]; // Visited vertices

// Function to find the vertex with minimum distance

int minDistance(int n) {

int min = INF, min\_index = -1;

for (int v = 0; v < n; v++) {

if (!visited[v] && dist[v] <= min) {

min = dist[v];

min\_index = v;

}

}

return min\_index;

}

// Dijkstra's algorithm

void dijkstra(int n, int src) {

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

dist[i] = INF;

visited[i] = 0;

}

dist[src] = 0;

for (int count = 0; count < n - 1; count++) {

int u = minDistance(n);

visited[u] = 1;

for (int v = 0; v < n; v++) {

if (!visited[v] && graph[u][v] && dist[u] != INF &&

dist[u] + graph[u][v] < dist[v]) {

dist[v] = dist[u] + graph[u][v];

}

}

}

// Print shortest distances

printf("Vertex\tDistance from Source %d\n", src);

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

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

}

}

int main() {

int n, src;

// Input: number of vertices

printf("Enter number of vertices: ");

scanf("%d", &n);

// Input: adjacency matrix

printf("Enter adjacency matrix (0 if no edge):\n");

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

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

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

// Input: source vertex

printf("Enter source vertex: ");

scanf("%d", &src);

// Run Dijkstra's algorithm

dijkstra(n, src);

return 0;

}