```
#include <iostream>

#include <vector>

#include <climits>

using namespace std;



const int MAX = 100;



void dijkstra(int graph[MAX][MAX], int n, int src) {

vector<int> dist(n, INT_MAX);

vector<bool> visited(n, false);



dist[src] = 0;



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

int u = -1;
// Find the unvisited node with the smallest distance

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

if (!visited[i] && (u == -1 || dist[i] < dist[u]))

u = i;

}



visited[u] = true;



// Update distances of adjacent nodes

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

if (graph[u][v] && !visited[v] && dist[u] + graph[u][v] < dist[v]) {

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

}

}

}



cout << "Shortest distances from source (" << src << "):\n";
```

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

cout << "To node " << i << " : " << (dist[i] == INT_MAX ? "Infinity" : to_string(dist[i])) << endl;

}

}



int main() {

int n, graph[MAX][MAX], src;



cout << "Enter the number of landmarks (nodes): ";

cin >> n;



cout << "Enter the adjacency matrix (enter 0 if no direct link exists):\n";

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

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

cin >> graph[i][j];
```

```
cout << "Enter the source landmark (0 to " << n - 1 << "): ";

cin >> src;



dijkstra(graph, n, src);



return 0;

}
```