#include <iostream>

#include <vector>

#include <limits.h>

using namespace std;

struct Edge

{

int source, destination, weight;

};

void bellmanFord(vector<Edge> &edges, int V, int E, int source)

{

vector<int> distance(V, INT\_MAX);

vector<int> predecessor(V, -1);

distance[source] = 0;

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

{

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

{

int u = edges[j].source;

int v = edges[j].destination;

int w = edges[j].weight;

if (distance[u] != INT\_MAX && distance[u] + w < distance[v])

{

distance[v] = distance[u] + w;

predecessor[v] = u;

}

}

}

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

{

if (i != source) //confuse

{

cout << "Shortest path from " << source << " to " << i << ": ";

// Build and print the shortest path

int current = i;

vector<int> path;

while (current != -1) // confusing

{

path.push\_back(current);

current = predecessor[current];

}

// Print the path in reverse order

for (int j = path.size() - 1; j >= 0; --j) // here path

{

cout << path[j];

if (j != 0)

{

cout << " -> ";

}

}

cout << " Distance = " << distance[i] << "\n";

}

}

}

int main()

{

int V, E;

cout << "Enter the number of vertices: ";

cin >> V;

cout << "Enter the number of edges: ";

cin >> E;

vector<Edge> edges(E);

cout << "Enter the edges (source, destination, weight):\n";

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

{

cin >> edges[i].source >> edges[i].destination >> edges[i].weight;

}

int source;

cout << "Enter the source vertex: ";

cin >> source;

bellmanFord(edges, V, E, source);

return 0;

}