// A C++ program for Bellman-Ford's single source

// shortest path algorithm.

#include <bits/stdc++.h>

// The main function that finds shortest distances from src to

// all other vertices using Bellman-Ford algorithm.  The function

// also detects negative weight cycle

void BellmanFord(struct Graph\* graph, int src)

{

    int V = graph->V;

    int E = graph->E;

    int dist[V];

    // Step 1: Initialize distances from src to all other vertices

    // as INFINITE

    for (int i = 0; i < V; i++)

        dist[i]   = INT\_MAX;

    dist[src] = 0;

    // Step 2: Relax all edges |V| - 1 times. A simple shortest

    // path from src to any other vertex can have at-most |V| - 1

    // edges

    for (int i = 1; i <= V-1; i++)

    {

        for (int j = 0; j < E; j++)

        {

            int u = graph->edge[j].src;

            int v = graph->edge[j].dest;

            int weight = graph->edge[j].weight;

            if (dist[u] != INT\_MAX && dist[u] + weight < dist[v])

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

        }

    }

    // Step 3: check for negative-weight cycles.  The above step

    // guarantees shortest distances if graph doesn't contain

    // negative weight cycle.  If we get a shorter path, then there

    // is a cycle.

    for (int i = 0; i < E; i++)

    {

        int u = graph->edge[i].src;

        int v = graph->edge[i].dest;

        int weight = graph->edge[i].weight;

        if (dist[u] != INT\_MAX && dist[u] + weight < dist[v])

            printf("Graph contains negative weight cycle");

    }

    printArr(dist, V);

    return;

}