#include <stdio.h>

#include <limits.h>

#define V 6 // Number of vertices in the graph

// Function to find the vertex with the minimum distance value,

// from the set of vertices not yet included in the shortest path tree.

int minDistance(int dist[], int sptSet[]) {

int min = INT\_MAX, min\_index;

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

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

min = dist[v];

min\_index = v;

}

}

return min\_index;

}

// Function to print the final shortest distance array.

void printSolution(int dist[]) {

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

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

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

}

// Function to implement Dijkstra's algorithm for a given graph.

void dijkstra(int graph[V][V], int src) {

int dist[V]; // The output array to store the shortest distance from the source vertex to each vertex.

int sptSet[V]; // sptSet[i] will be true if the vertex i is included in the shortest path tree.

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

dist[i] = INT\_MAX;

sptSet[i] = 0;

}

dist[src] = 0;

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

int u = minDistance(dist, sptSet);

sptSet[u] = 1;

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

if (!sptSet[v] && graph[u][v] && dist[u] != INT\_MAX &&

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

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

}

}

}

printSolution(dist);

}

int main() {

int graph[V][V] = {

{7, 4, 0, 3, 0, 0},

{4, 0, 8, 0, 0, 0},

{0, 8, 0, 7, 0, 4},

{2, 0, 7, 0, 9, 14},

{0, 0, 10, 9, 0, 10},

{4, 0, 4, 14, 10, 0}

};

dijkstra(graph, 0); // Calculate shortest paths from vertex 0

return 0;

}