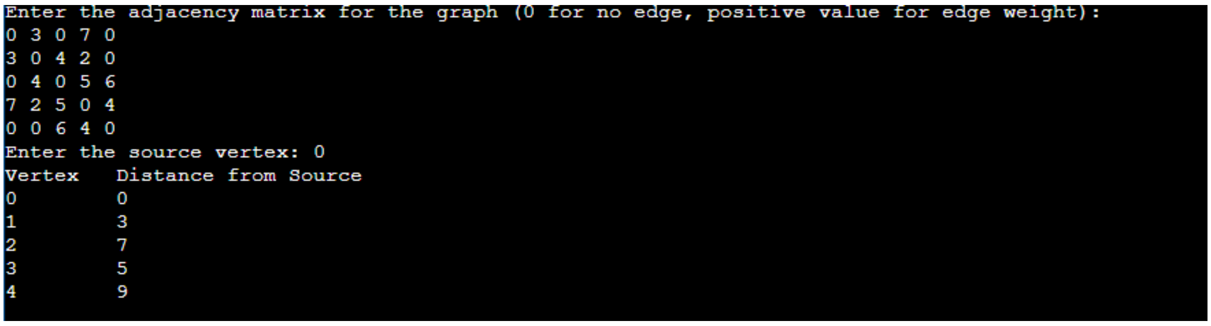
```
#include <stdio.h>

#include <stdbool.h>

#include <limits.h>

#define V 5

int minDistance(int dist[], bool 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; }

void printSolution(int dist[]) {

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

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

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

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

    int dist[V];

    bool sptSet[V];

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

        dist[i] = INT_MAX, sptSet[i] = false;

    dist[src] = 0;

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

        int u = minDistance(dist, sptSet);

        sptSet[u] = true;

       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];

    printf("Enter the adjacency matrix for the graph (0 for no edge, positive value for edge
weight):\n");
for (int i = 0; i < V; i++)
    for (int j = 0; j < V; j++)
        scanf("%d", &graph[i][j]);
int source;
printf("Enter the source vertex: ");
scanf("%d", &source);
dijkstra(graph, source);
return 0; }
```



```
Enter the adjacency matrix for the graph (0 for no edge, positive value for edge weight):
0 3 0 7 0
3 0 4 2 0
0 4 0 5 6
7 2 5 0 4
0 0 6 4 0
Enter the source vertex: 0
Vertex   Distance from Source
0        0
1        3
2        7
3        5
4        9
```