```c
#include <stdio.h>
#include <stdlib.h>

#define INF 1000000
#define V 6

typedef struct {
    int vertex;
    int weight;
} Edge;

typedef struct {
    int vertex;
    int distance;
} Node;

// Function to print the shortest path
void printPath(int path[], int start, int end) {
    printf("Shortest path from %d to %d: ", start, end);
    for (int i = 0; i < V; i++) {
        printf("%d ", path[i]);
    }
    printf("\n");
}

// Function to implement Dijkstra's algorithm
void dijkstra(int graph[V][V], int start) {
    int distance[V];
    int path[V];
    int visited[V];

    // Initialize distance and path arrays
    for (int i = 0; i < V; i++) {
        distance[i] = INF;
        path[i] = -1;
        visited[i] = 0;
    }
    distance[start] = 0;

    // Loop until all nodes are visited
    for (int i = 0; i < V; i++) {
        int min = INF;
        int min_index = -1;

        for (int j = 0; j < V; j++) {
            if (!visited[j] && distance[j] < min) {
                min = distance[j];
                min_index = j;
            }
        }

        visited[min_index] = 1;

        for (int j = 0; j < V; j++) {
            if (!visited[j] && graph[min_index][j] != 0) {
                if (distance[min_index] + graph[min_index][j] < distance[j]) {
                    distance[j] = distance[min_index] + graph[min_index][j];
                    path[j] = min_index;
                }
            }
        }
    }

    printPath(path, start, V - 1);
}

int main() {
    int graph[V][V] = {
        {0, 4, 0, 0, 0, 0},
        {4, 0, 8, 0, 0, 0},
        {0, 8, 0, 7, 0, 4},
        {0, 0, 7, 0, 9, 14},
        {0, 0, 0, 9, 0, 10},
        {0, 0, 4, 14, 10, 0}
    };
    dijkstra(graph, 0);

    return 0;
}
```