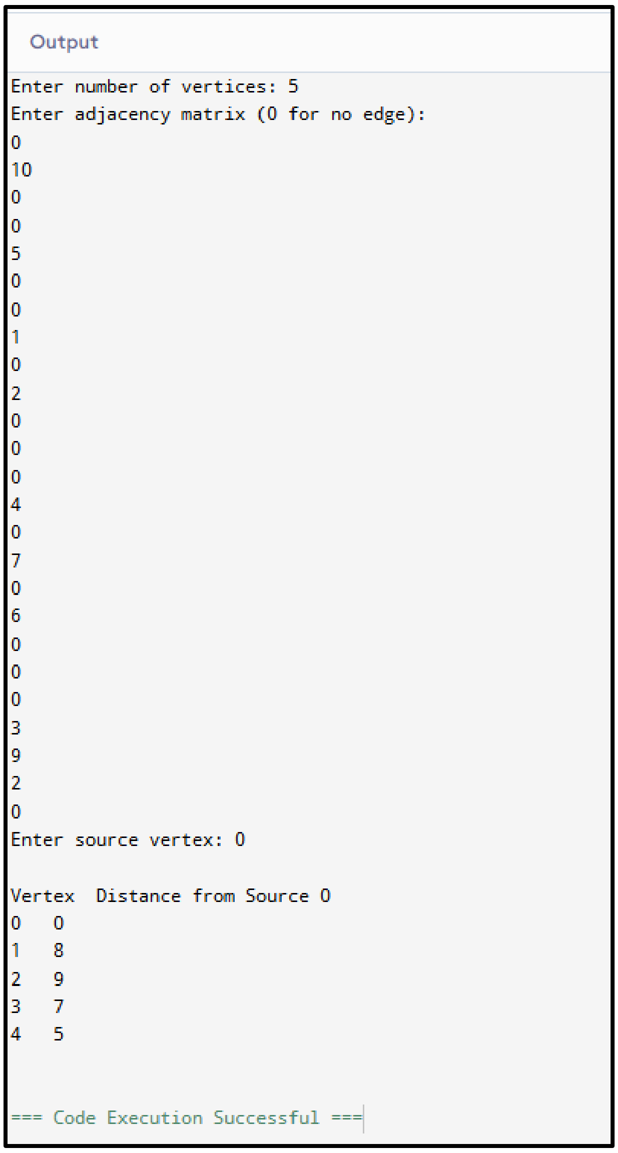
```c
#include <stdio.h>
#include <limits.h>
#define MAX 10
#define INF 9999
int minDist(int dist[], int visited[], int n) {
    int min = INF, index = -1;
    for (int i = 0; i < n; i++) {
        if (!visited[i] && dist[i] < min) {
            min = dist[i];
            index = i;
        }
    }
    return index;
}
void dijkstra(int graph[MAX][MAX], int n, int src) {
    int dist[MAX], visited[MAX];
    for (int i = 0; i < n; i++) {
        dist[i] = INF;
        visited[i] = 0;
    }
    dist[src] = 0;
    for (int count = 0; count < n - 1; count++) {
        int u = minDist(dist, visited, n);
        if (u == -1) break;
        visited[u] = 1;
        for (int v = 0; v < n; v++) {
            if (!visited[v] && graph[u][v] && dist[u] + graph[u][v]
< dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }
    printf("\nVertex\tDistance from Source %d\n", src);
    for (int i = 0; i < n; i++) {
        if (dist[i] == INF)
            printf("%d\tINF\n", i);
        else
            printf("%d\t%d\n", i, dist[i]);
    }
}
int main() {
    int n, graph[MAX][MAX], src;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix (0 for no edge):\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);
    printf("Enter source vertex: ");
    scanf("%d", &src);
    dijkstra(graph, n, src);
    return 0;
}
```



Output

```
Enter number of vertices: 5
Enter adjacency matrix (0 for no edge):
0
10
0
0
5
0
0
1
0
2
0
0
0
4
0
7
0
6
0
0
0
3
9
2
0
Enter source vertex: 0

Vertex  Distance from Source 0
0   0
1   8
2   9
3   7
4   5


=== Code Execution Successful ===
```