Exp_12\Dijkastra.c

```c
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>

#define MAX 100
#define INF INT_MAX

int minDistance(int dist[], bool visited[], int n) {
    int min = INF, minIndex;
    for (int i = 0; i < n; i++) {
        if (!visited[i] && dist[i] < min) {
            min = dist[i];
            minIndex = i;
        }
    }
    return minIndex;
}

void dijkstra(int graph[MAX][MAX], int n, int src) {
    int dist[MAX];
    bool visited[MAX];

    for (int i = 0; i < n; i++) {
        dist[i] = INF;
        visited[i] = false;
    }

    dist[src] = 0;

    for (int count = 0; count < n - 1; count++) {
        int u = minDistance(dist, visited, n);
        visited[u] = true;

        for (int v = 0; v < n; v++) {
            if (!visited[v] && graph[u][v] && dist[u] != INF &&
                dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }

    printf("Vertex\t\tDistance from Source (%d)\n", src);
    for (int i = 0; i < n; i++) {
        printf("%d\t\t", i);
        if (dist[i] == INF)
            printf("INF\n");
        else
            printf("%d\n", dist[i]);
    }
}

```

```c
int main() {
    int n, src;
    int graph[MAX][MAX];

    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;
}
```