Name	**: Mulya Fajar Ningsih Alwi**
Student ID	**: 001202000101**
Major/Class	**: CIT/3**
Batch	**: 2020**

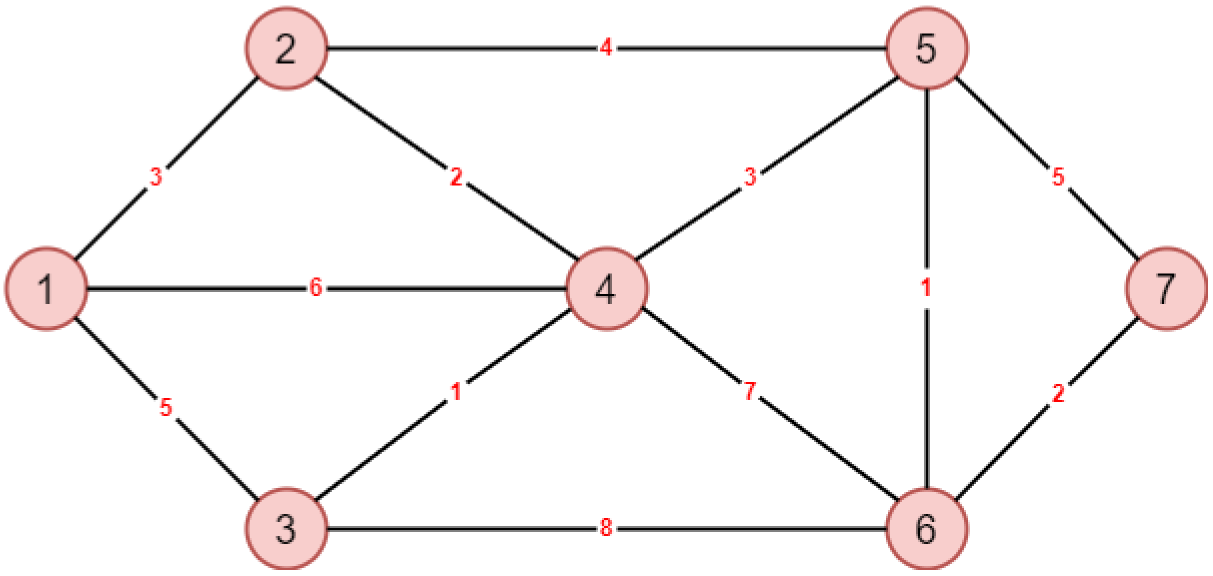
Reading Assignment 1: Dijkstra Algorithm

Dijkstra's algorithm, (according to its inventor Edsger Dijkstra), is an algorithm used to solve the shortest path problem for a directed graph.

Dijkstra's algorithm works by creating a path to one optimal node at each step. So, at step n there are at least n nodes for which we already know the shortest path. Dijkstra's algorithm steps can be done with the following steps:

1. Determine which point will be the starting node, then give the distance value for the first node to the nearest node one by one, Dijkstra will develop the search from one point to another and to the next point step by step.
2. Assign a value (distance) for each point to another point, then set a value of 0 to the starting node and an infinity value to the other nodes.
3. Set all unvisited nodes and set the start node as “Start node”.
4. From the start node, consider the neighboring nodes that have not been visited and calculate their distance from the start point. If this distance is smaller than the previous distance (which has been recorded previously) delete the old data, and re-save the distance data with the new distance.
5. When we are done considering each distance to the neighboring node, mark the node that has been visited as “Node visited”. The node that is passed will never be checked again. The distance stored is the last distance with the least value.
6. Set “Node not yet visited” with the minimum distance (from start node) as the next “Start node” and repeat step 5.

Below is an example of a case study related to Dijkstra's algorithm along with the program using the C language:



Rosie's house is at point 1, if Rosie wants to go to the bookstore, Rosie must be at point 7 and the points that Rosie can pass are points 2, 3, 4, 5, and 6. Find the shortest distance Rosie can take to get to the bookstore faster!

Analysis of the distance of each path:

- 1 → 2 = 3 (the opposite applies, 1 → 2 = 3 same with 2 → 1 = 3)
- 1 → 3 = 5
- 1 → 4 = 6
- 2 → 4 = 2
- 2 → 5 = 4
- 3 → 4 = 1
- 3 → 6 = 8
- 4 → 5 = 3
- 4 → 6 = 7
- 5 → 6 = 1
- 5 → 7 = 5
- 6 → 7 = 2

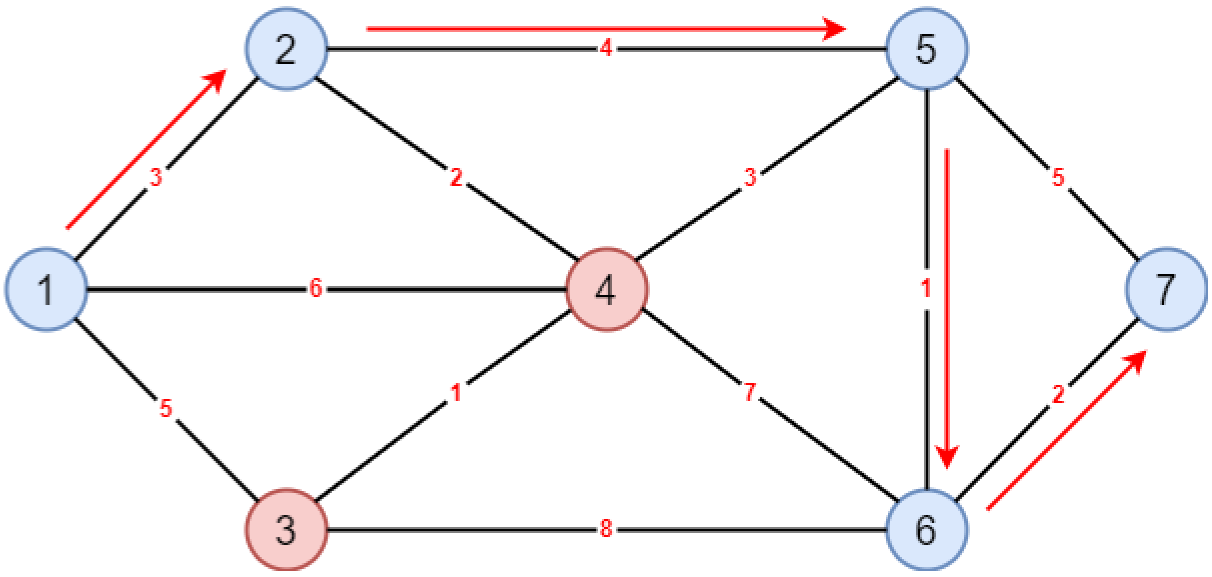
Analysis of the distance of each path in a table form:

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	3	5	6	0	0	0
2	3	0	0	2	4	0	0
3	5	0	0	1	0	8	0
4	6	2	1	0	3	7	0
5	0	4	0	3	0	1	5
6	0	0	8	7	1	0	2
7	0	0	0	0	5	2	0

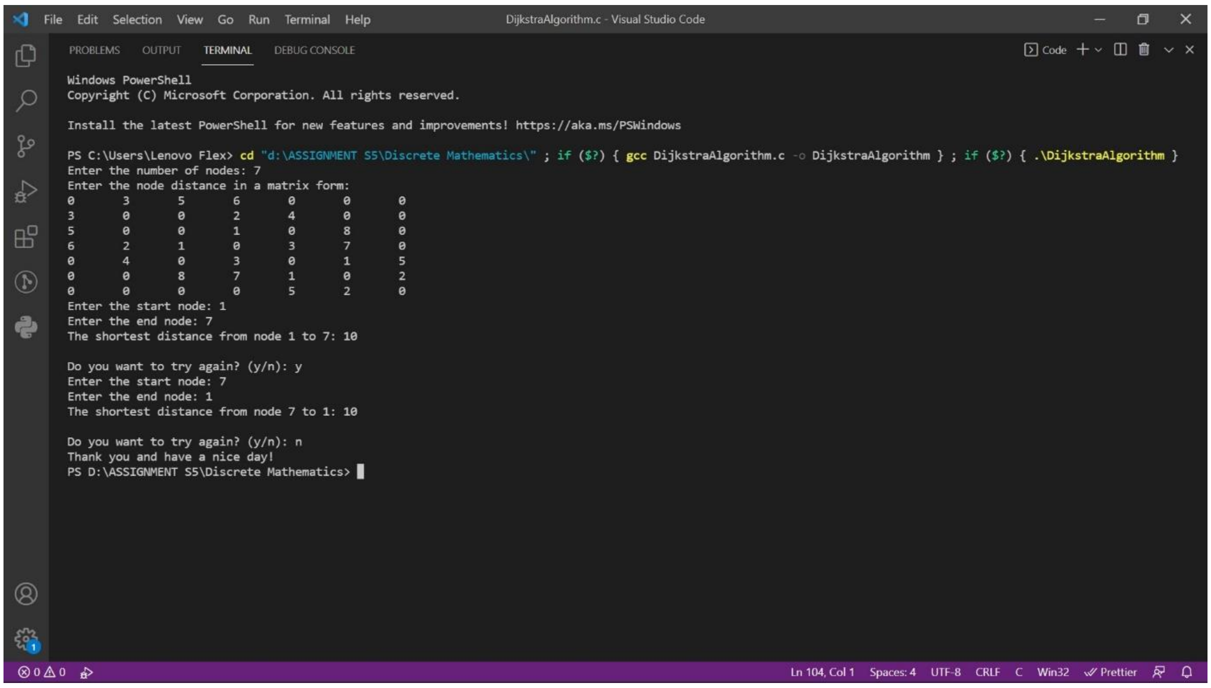
Analysis of the distance of each path in a matrix form:

$$\begin{matrix} \mathbf{0} & \mathbf{3} & \mathbf{5} & \mathbf{6} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{3} & \mathbf{0} & \mathbf{0} & \mathbf{2} & \mathbf{4} & \mathbf{0} & \mathbf{0} \\ \mathbf{5} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{8} & \mathbf{0} \\ \mathbf{6} & \mathbf{2} & \mathbf{1} & \mathbf{0} & \mathbf{3} & \mathbf{7} & \mathbf{0} \\ \mathbf{0} & \mathbf{4} & \mathbf{0} & \mathbf{3} & \mathbf{0} & \mathbf{1} & \mathbf{5} \\ \mathbf{0} & \mathbf{0} & \mathbf{8} & \mathbf{7} & \mathbf{1} & \mathbf{0} & \mathbf{2} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{5} & \mathbf{2} & \mathbf{0} \end{matrix}$$

By using direct analysis through graphs, we can find out the shortest distance that Rosie can take to the bookstore is 1 → 2 → 5 → 6 → 7 = 10.

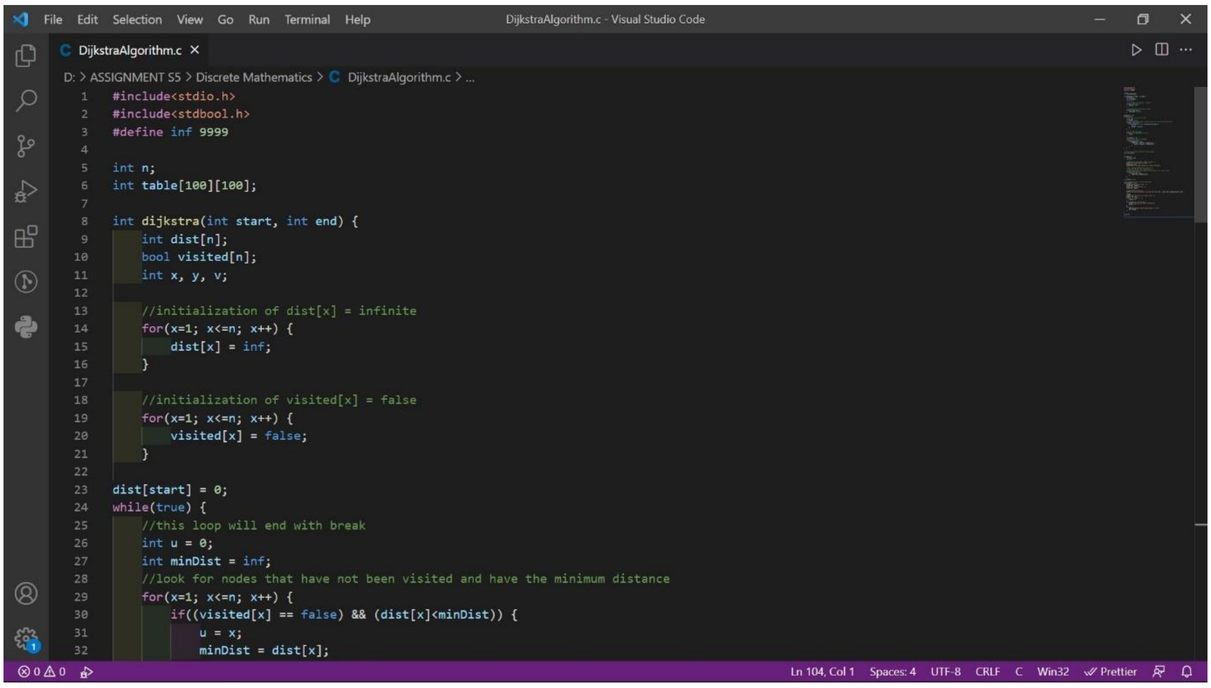


Using the program will also gain the same result, the shortest distance from Rosie's house to the bookstore or node 1 to 7 = 10.

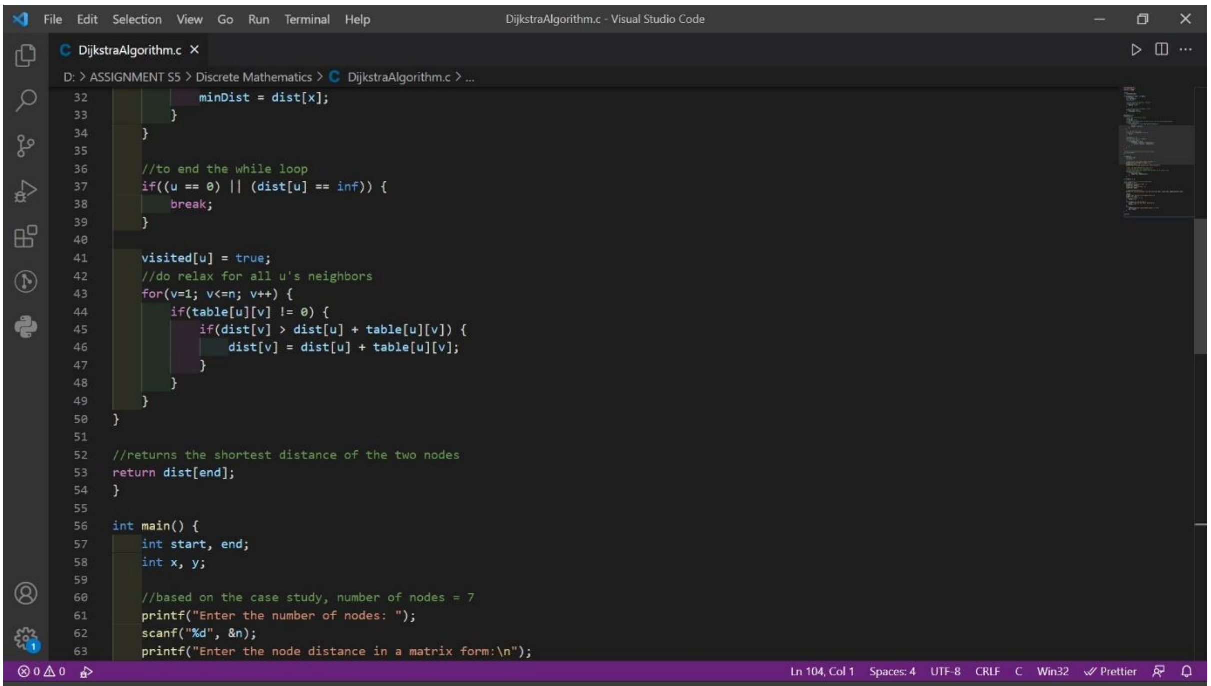


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Lenovo Flex> cd "d:\ASSIGNMENT S5\Discrete Mathematics\" ; if ($?) { gcc DijkstraAlgorithm.c -o DijkstraAlgorithm } ; if ($?) { .\DijkstraAlgorithm }
Enter the number of nodes: 7
Enter the node distance in a matrix form:
0       3       5       6       0       0       0
3       0       0       2       4       0       0
5       0       0       1       0       8       0
6       2       1       0       3       7       0
0       4       0       3       0       1       5
0       0       8       7       1       0       2
0       0       0       0       5       2       0
Enter the start node: 1
Enter the end node: 7
The shortest distance from node 1 to 7: 10

Do you want to try again? (y/n): y
Enter the start node: 7
Enter the end node: 1
The shortest distance from node 7 to 1: 10

Do you want to try again? (y/n): n
Thank you and have a nice day!
PS D:\ASSIGNMENT S5\Discrete Mathematics> █
```

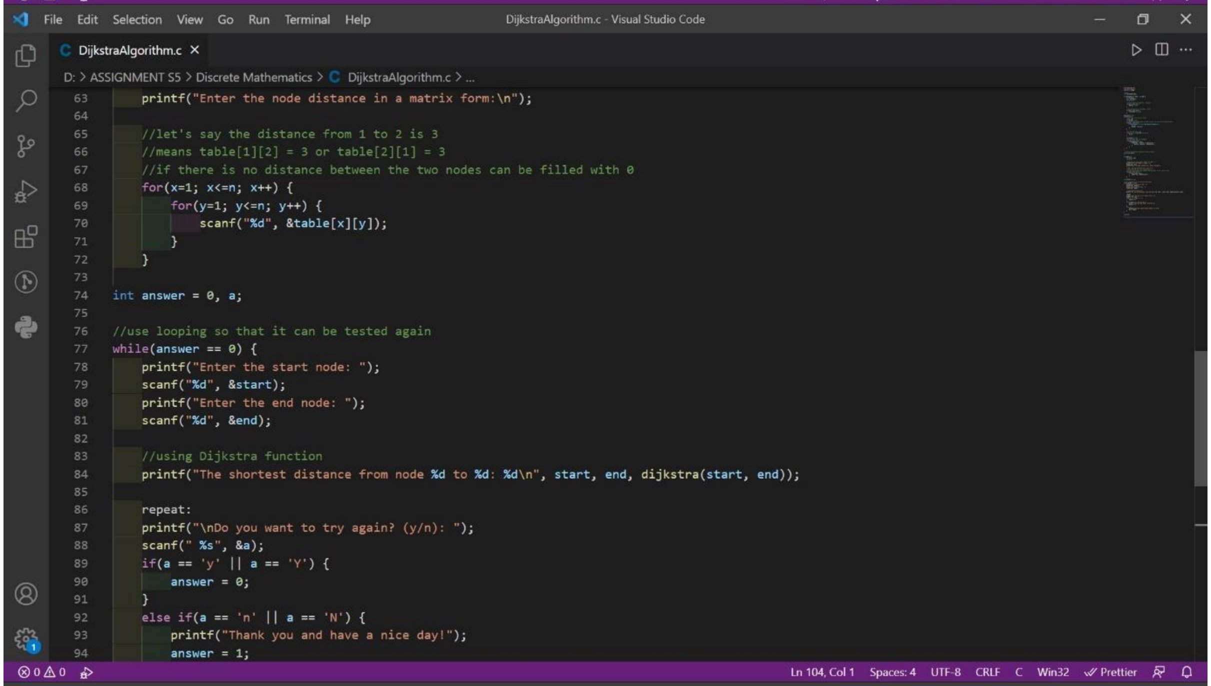
Below is the program code along with the documentation.



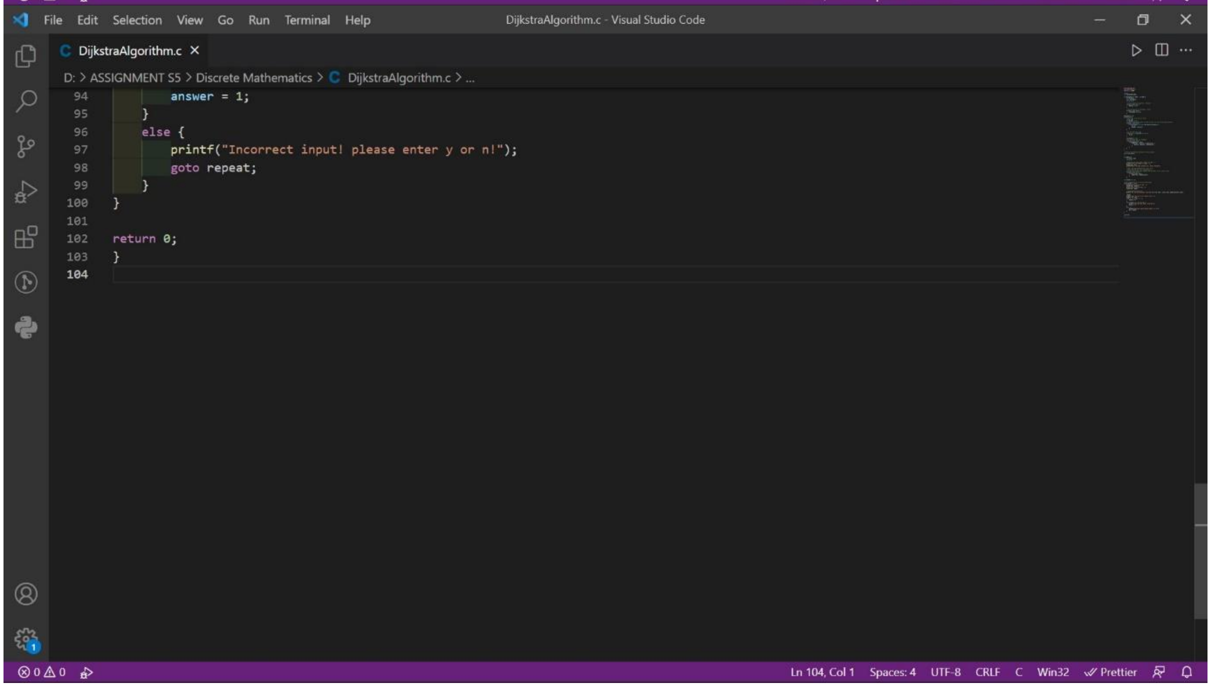
```c
#include<stdio.h>
#include<stdbool.h>
#define inf 9999

int n;
int table[100][100];

int dijkstra(int start, int end) {
    int dist[n];
    bool visited[n];
    int x, y, v;

    //initialization of dist[x] = infinite
    for(x=1; x<=n; x++) {
        dist[x] = inf;
    }

    //initialization of visited[x] = false
    for(x=1; x<=n; x++) {
        visited[x] = false;
    }

dist[start] = 0;
while(true) {
    //this loop will end with break
    int u = 0;
    int minDist = inf;
    //look for nodes that have not been visited and have the minimum distance
    for(x=1; x<=n; x++) {
        if((visited[x] == false) && (dist[x]<minDist)) {
            u = x;
            minDist = dist[x];
```


DijkstraAlgorithm.c - Visual Studio Code
D: > ASSIGNMENT S5 > Discrete Mathematics > DijkstraAlgorithm.c > ...
Ln 104, Col 1 Spaces: 4 UTF-8 CRLF C Win32 Prettier

```c
            minDist = dist[x];
        }
    }

    //to end the while loop
    if((u == 0) || (dist[u] == inf)) {
        break;
    }

    visited[u] = true;
    //do relax for all u's neighbors
    for(v=1; v<=n; v++) {
        if(table[u][v] != 0) {
            if(dist[v] > dist[u] + table[u][v]) {
                dist[v] = dist[u] + table[u][v];
            }
        }
    }
}

//returns the shortest distance of the two nodes
return dist[end];
}

int main() {
    int start, end;
    int x, y;

    //based on the case study, number of nodes = 7
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    printf("Enter the node distance in a matrix form:\n");
```


DijkstraAlgorithm.c - Visual Studio Code
D: > ASSIGNMENT S5 > Discrete Mathematics > DijkstraAlgorithm.c > ...
Ln 104, Col 1 Spaces: 4 UTF-8 CRLF C Win32 Prettier

```c
    printf("Enter the node distance in a matrix form:\n");

    //let's say the distance from 1 to 2 is 3
    //means table[1][2] = 3 or table[2][1] = 3
    //if there is no distance between the two nodes can be filled with 0
    for(x=1; x<=n; x++) {
        for(y=1; y<=n; y++) {
            scanf("%d", &table[x][y]);
        }
    }

int answer = 0, a;

//use looping so that it can be tested again
while(answer == 0) {
    printf("Enter the start node: ");
    scanf("%d", &start);
    printf("Enter the end node: ");
    scanf("%d", &end);

    //using Dijkstra function
    printf("The shortest distance from node %d to %d: %d\n", start, end, dijkstra(start, end));

    repeat:
    printf("\nDo you want to try again? (y/n): ");
    scanf(" %s", &a);
    if(a == 'y' || a == 'Y') {
        answer = 0;
    }
    else if(a == 'n' || a == 'N') {
        printf("Thank you and have a nice day!");
        answer = 1;
```


DijkstraAlgorithm.c - Visual Studio Code
D: > ASSIGNMENT S5 > Discrete Mathematics > DijkstraAlgorithm.c > ...
Ln 104, Col 1 Spaces: 4 UTF-8 CRLF C Win32 Prettier

```c
        answer = 1;
    }
    else {
        printf("Incorrect input! please enter y or n!");
        goto repeat;
    }
}

return 0;
}

```