https://github.com/aniketjain4004/week-1-b

```
#include <stdio.h> int main() {     int rows,
cols, i, j;     printf("Enter rows and columns:
");     scanf("%d %d", &rows, &cols);     int
matrix[rows][cols], transpose[cols][rows];
printf("Enter the matrix elements:\n");     for(i
= 0; i < rows; i++)         for(j = 0; j < cols;
j++)             scanf("%d", &matrix[i][j]);
    // Transpose logic     for(i = 0; i <
rows; i++)         for(j = 0; j < cols;
j++)             transpose[j][i] =
matrix[i][j];     printf("Transposed
matrix:\n");     for(i = 0; i < cols; i++)
{         for(j = 0; j < rows; j++)
printf("%d ", transpose[i][j]);
printf("\n");
    }
return 0;
}
```

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define MAX_WORDS 100
#define MAX_LEN   50
 int main()
{
    char para[1000], word[MAX_WORDS][MAX_LEN];
int freq[MAX_WORDS] = {0}, i = 0, count = 0;
printf("Enter a paragraph:\n");     fgets(para,
1000, stdin);

    char *token = strtok(para, " ,.-\n");
while(token) {
        // Lowercase         for(int k = 0; token[k]; k++) token[k]
= tolower(token[k]);          int found = 0;          for(int j = 0;
j < count; j++) {             if(strcmp(word[j], token) == 0) {
freq[j]++;                 found = 1; break;
            }         }
if(!found) {
strcpy(word[count], token);
freq[count++] = 1;
        }         token = strtok(NULL,
" ,.-\n");
    }      printf("Word frequencies:\n");     for(i = 0; i < count;
i++) printf("%s: %d\n", word[i], freq[i]);     return 0;
}
```

```
# Dijkstra's Algorithm in C

## Overview

This C program implements **Dijkstra's shortest path algorithm** for a
directed or undirected weighted graph using an adjacency matrix. It
calculates the shortest distance from a given starting node to all other
nodes in the graph and displays the distance and paths.

## Features

- Accepts a custom number of vertices from the user.
- Reads the adjacency matrix (edge weights) from user input.
- Allows the user to set the starting node.
- Outputs the shortest distance and the shortest path from the starting
  node to every other node.
```

~~~~markdown
## How to Compile
 ```bash gcc dijkstra.c -o
dijkstra
```

## How to Run

```bash
./dijkstra
```
Or, in Windows:
```bat dijkstra.exe
```

## Input Format

1. Number of vertices (n)
2. n × n adjacency matrix; use 0 for no edge between i and j
3. Starting node (index starts from 0)

## Example Session

```
Enter no. of vertices: 4

Enter the adjacency matrix:
0 1 3 0
1 0 1 7
3 1 0 2
0 7 2 0

Enter the starting node: 0

Distance of node1=1 Path=1<-0
Distance of node2=2
Path=2<-1<-0
~~~~

```
Distance of node3=4
Path=3<-2<-1<-0
```

## Example Adjacency Matrix

For the following graph (4 vertices):

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 3 | 0 |
| 1 | 1 | 0 | 1 | 7 |
| 2 | 3 | 1 | 0 | 2 |
| 3 | 0 | 7 | 2 | 0 |

## Notes

- The program treats 0 as "no path" (except on the diagonal).
- Maximum allowed vertices: 10 (change `MAX` macro for larger graphs).
- Node indices start from 0.
- The shortest path and its total distance for every node (except the starting one) are displayed.

## License

This project is open-source and intended for educational use.