Q2-

int rows, cols;

printf("Enter number of rows: ");

scanf("%d", &rows);

printf("Enter number of columns: ");

scanf("%d", &cols);

int matrix[rows][cols], transpose[cols][rows];

// Input matrix

printf("Enter elements of the matrix:\n");

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

printf("Element [%d][%d]: ", i, j);

scanf("%d", &matrix[i][j]);

}

}

// Transpose

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

transpose[j][i] = matrix[i][j];

}

}

// Print transposed matrix

printf("\nTranspose of the matrix:\n");

for (int i = 0; i < cols; i++) {

for (int j = 0; j < rows; j++) {

printf("%d ", transpose[i][j]);

}

printf("\n");

}

return 0;

}

Q3-

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX\_WORDS 1000

#define MAX\_WORD\_LEN 50

typedef struct {

char word[MAX\_WORD\_LEN];

int count;

} WordFreq;

void toLowerCase(char\* str) {

for (int i = 0; str[i]; i++)

str[i] = tolower(str[i]);

}

int main() {

char paragraph[1000];

WordFreq freq[MAX\_WORDS];

int wordCount = 0;

printf("Enter a paragraph:\n");

fgets(paragraph, sizeof(paragraph), stdin);

char \*token = strtok(paragraph, " ,.-\n");

while (token != NULL) {

toLowerCase(token);

int found = 0;

for (int i = 0; i < wordCount; i++) {

if (strcmp(freq[i].word, token) == 0) {

freq[i].count++;

found = 1;

break;

}

}

if (!found && wordCount < MAX\_WORDS) {

strcpy(freq[wordCount].word, token);

freq[wordCount].count = 1;

wordCount++;

}

token = strtok(NULL, " ,.-\n");

}

printf("\nWord Frequencies:\n");

for (int i = 0; i < wordCount; i++) {

printf("%s: %d\n", freq[i].word, freq[i].count);

}

return 0;

}

Q4- git clone

https://github.com/Maximetinu/Dijkstra-C.git

cd Dijkstra-C