HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

ASSIGNMENT 1

Name	: Muhammed Said
Surname	: Kaya
Student Number	: 21627428
Section	: 02
Subject	: Data Structures and Algorithms
Programming Language	: C
Advisor	: R.A. Alaettin UÇAN

1.INTRODUCTION / AIM

In this experiment , People are expected to gain knowledge on C language including read-write files, arrays , matrices , recursion and dynamic memory allocation.

2.PROBLEM

To find subMatrix which is divisible of 5 after multiplying with keyMatrix, in the mapMatrix.

3.HOW TO SOLVE PROBLEM (Summary Of Code)

According to size of keyMatrix , initial coordinates of subMatrix in the mapMatrix are determined. (For instance , size of keyMatrix is 3x3 so initial coordinates 1,1 but if it is 5x5 , initial coordinates 2,2) . According to mod of multiplication of subMatrix and keyMatrix , coordinates are changed (For instance mod of multiplication is 3 , after that index are going to be 1,4) Then again multiplication is done between new subMatrix and keyMatrix until mod of multiplication of subMatrix and keyMatrix is divisible by 5.

4.CONTENT / EXPLANATION

❶ In main.c :

1) argv[1] element is split and two numbers are assigned to two different Integer values named widthOfMapMatrix and heightOfMapMatrix.

```
int widthOfMapMatrix , heightOfMapMatrix , splitCounterForAssignHeightAndWidthOfMapMatrix = 0 ;
 // Split part
char *spliter = strtok( argv[1] , "x" )
while( spliter != NULL){

    if  (splitCounterForAssignHeightAndWidthOfMapMatrix == 0)    { heightOfMapMatrix   = atoi(spliter); }
    else                                                         { widthOfMapMatrix    = atoi(spliter); }

    splitCounterForAssignHeightAndWidthOfMapMatrix++;
    spliter = strtok(NULL,"x");
}
```

2) argv[2] element is assigned to an Integer value named sizeOfKeyMatrix.

```
int   sizeOfKeyMatrix = atoi(argv[2]) ;
```

3) argv[3] and argv[4] elements (the names of the documents containig mapMatrix and keyMatrix) are assigned to char pointers named textNameOfMapMatrix and textNameOfKeyMatrix.

```
char *textNameOfMapMatrix = argv[3] , *textNameOfKeyMatrix = argv[4] ;
```

4) The document ,that the result is going to be written, is opened by using argv[5].

```
FILE *outputTxt = fopen( argv[5], "w");
```

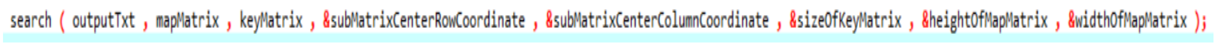
5) 2-D dynamic arrays named mapMatrix and keyMatrix are created by using the widthOfMapMatrix , heightOfMapMatrix and sizeOfKeyMatrix Integer values.

```
int  **mapMatrix , **keyMatrix ,counter;
// Part Of Creating Dynamic Arrays whose name is mapMatrix and keyMatrix  ,  according to size which are heightOfMapMatrix , widthOfMapMatrix and sizeOfKeyMatrix
mapMatrix = (int **) malloc ( sizeof(int *) *heightOfMapMatrix );              // mapMatrix malloc Part with Dynamic Memory
for(counter=0; counter < heightOfMapMatrix ; counter++)     {           mapMatrix[counter] = (int *) malloc ( sizeof(int) *widthOfMapMatrix);            }

keyMatrix = (int **) malloc ( sizeof (int *) *sizeOfKeyMatrix);       // keyMatrix malloc Part with Dynamic Memory
for(counter=0; counter < sizeOfKeyMatrix ; counter++){          keyMatrix[counter] = (int *) malloc ( sizeof(int) * sizeOfKeyMatrix);          }
```

6) The functions named setKeyMatrix and setMapMatrix, from file named functions.h, are run to fill the 2-dimensional dynamic arrays named keyMatrix and mapMatrix.

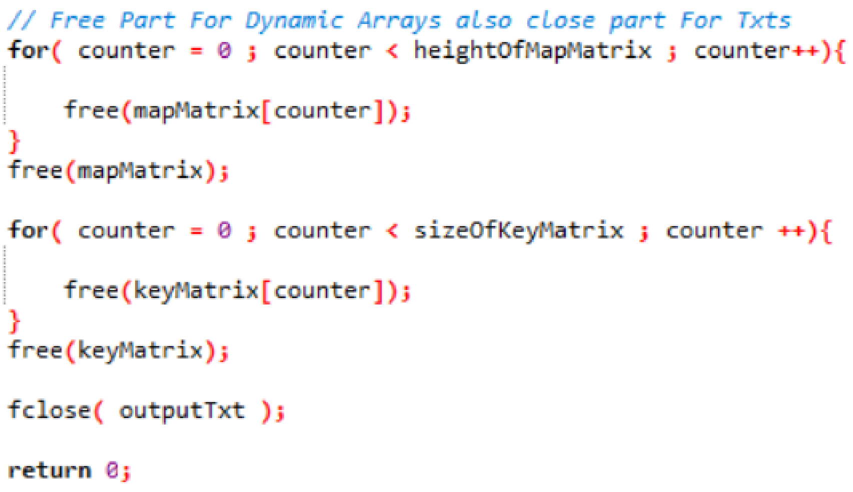
```
setMapMatrix( textNameOfMapMatrix , mapMatrix , &widthOfMapMatrix         );     // This function fiil the mapMatrix according to mapmatrix.txt
setKeyMatrix( textNameOfKeyMatrix , keyMatrix , &sizeOfKeyMatrix          );     // This function fill the keyMatrix according to keymatrix.txt
```

7) **The two-dimensional dynamic arrays mapMatrix and keyMatrix are sent to a function named search , that from file named functions.h also runs by recursive , for finding the result.**



```
search ( outputTxt , mapMatrix , keyMatrix , &subMatrixCenterRowCoordinate , &subMatrixCenterColumnCoordinate , &sizeOfKeyMatrix , &heightOfMapMatrix , &widthOfMapMatrix );
```

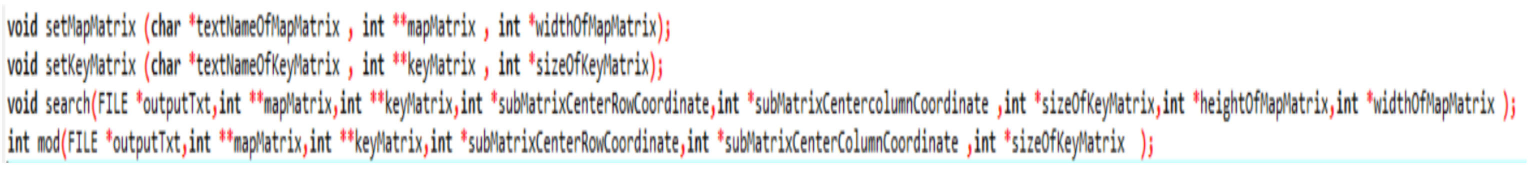
8) **The dynamic arrays named mapMatrix and keyMatrix are deleted from memory by using free and output file is closed by using fclose.**



```
// Free Part For Dynamic Arrays also close part For Txts
for( counter = 0 ; counter < heightOfMapMatrix ; counter++){

    free(mapMatrix[counter]);
}
free(mapMatrix);

for( counter = 0 ; counter < sizeOfKeyMatrix ; counter ++){

    free(keyMatrix[counter]);
}
free(keyMatrix);

fclose( outputTxt );

return 0;
```

❷ In functions.h :

- ⦿ There are 4 functions named setMapMatrix , setKeyMatrix , search and mod in the file called functions.h . In main.c ,These functions are accessed by using #include " functions.h"



```
void setMapMatrix (char *textNameOfMapMatrix , int **mapMatrix , int *widthOfMapMatrix);
void setKeyMatrix (char *textNameOfKeyMatrix , int **keyMatrix , int *sizeOfKeyMatrix);
void search(FILE *outputTxt,int **mapMatrix,int **keyMatrix,int *subMatrixCenterRowCoordinate,int *subMatrixCentercolumnCoordinate ,int *sizeOfKeyMatrix,int *heightOfMapMatrix,int *widthOfMapMatrix );
int mod(FILE *outputTxt,int **mapMatrix,int **keyMatrix,int *subMatrixCenterRowCoordinate,int *subMatrixCenterColumnCoordinate ,int *sizeOfKeyMatrix  );
```

❸ In setMapMatrix.c :

1) The name of the documents containig mapMatrix was assigned to char pointer named textNameOfMapMatrix in main.c . The file is opened by using this char pointer .

```
FILE *textOfMapMatrix = fopen( textNameOfMapMatrix , "r");    // Opening Part
```

2) When reading the file , The values obtained are placed in the 2D array named mapMatrix.

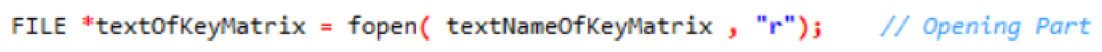
```
do{

    fscanf( textOfMapMatrix , "%s" , tempArray );     // Reading Part
    spliter = strtok(tempArray, " ");

    while (spliter != NULL) {

        mapMatrix[counter][secondCounter] = atoi(spliter);    // Assigning Part

        secondCounter++;
        spliter = strtok(NULL, " ");
    }
    if( secondCounter == (*widthOfMapMatrix) ){

        counter++;
        secondCounter = 0;
    }
}while( !feof(textOfMapMatrix) );
```

3) The dynamic array called tempArray, which is used for split when reading the file, is cleared from memory by using free and The file , that includes mapMatrix , is closed by using fclose.

```
free( tempArray );            // Free Part
fclose( textOfMapMatrix );    // mapmatrix.txt  Closing Part
```

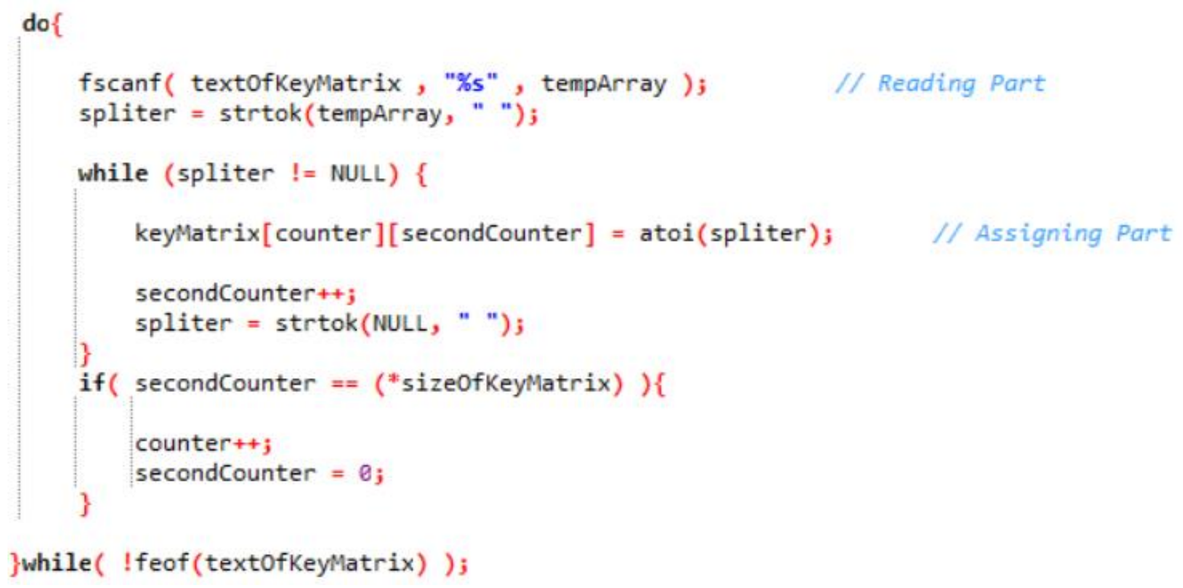
❹ In setKeyMatrix.c :

1) The name of the documents containig keyMatrix was assigned to char pointer named textNameOfKeyMatrix in main.c . The file is opened by using this char pointer .



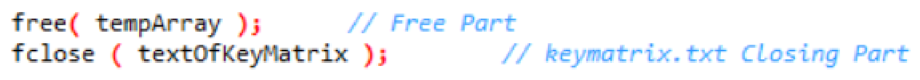
```
FILE *textOfKeyMatrix = fopen( textNameOfKeyMatrix , "r");    // Opening Part
```

2) When reading the file , The values obtained are placed in the 2D array named keyMatrix.



```
do{

    fscanf( textOfKeyMatrix , "%s" , tempArray );          // Reading Part
    spliter = strtok(tempArray, " ");

    while (spliter != NULL) {

        keyMatrix[counter][secondCounter] = atoi(spliter);       // Assigning Part

        secondCounter++;
        spliter = strtok(NULL, " ");
    }
    if( secondCounter == (*sizeOfKeyMatrix) ){

        counter++;
        secondCounter = 0;
    }

}while( !feof(textOfKeyMatrix) );
```

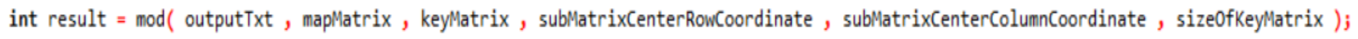
3) The dynamic array called tempArray, which is used for split when reading the file, is cleared from memory by using free and The file , that includes keyMatrix , is closed by using fclose.



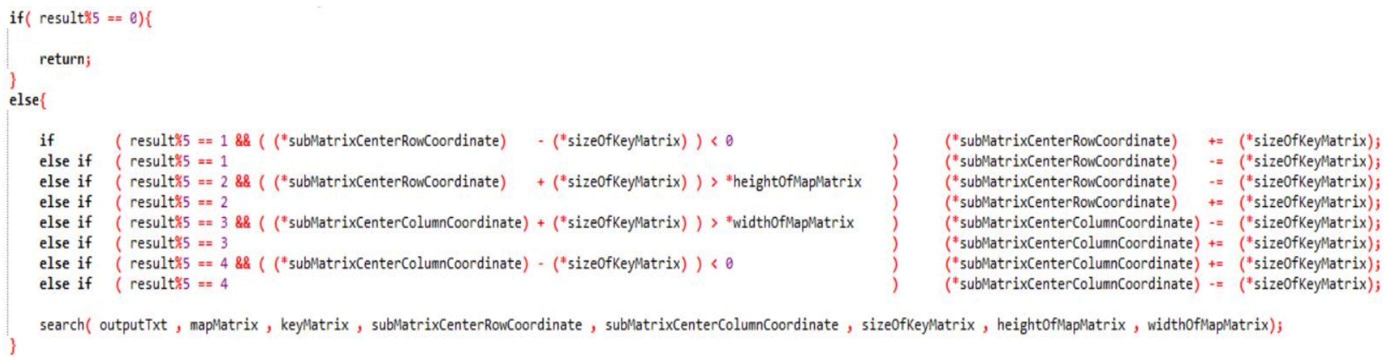
```
free( tempArray );       // Free Part
fclose ( textOfKeyMatrix );        // keymatrix.txt Closing Part
```

❺ In search.c :

⦿ ***How Recursion function works ?*** : While First running this function in main.c , The index values of the mapMatrix center, determined by the size of the keyMatrix at main.c, come to this function as an argument also the file that was opened for outputs come as an argument named outputTxt . For other running , according to the value returned by the mod function, index values are changed and the function named search is called with new index values until the function returns a divisible value of 5.



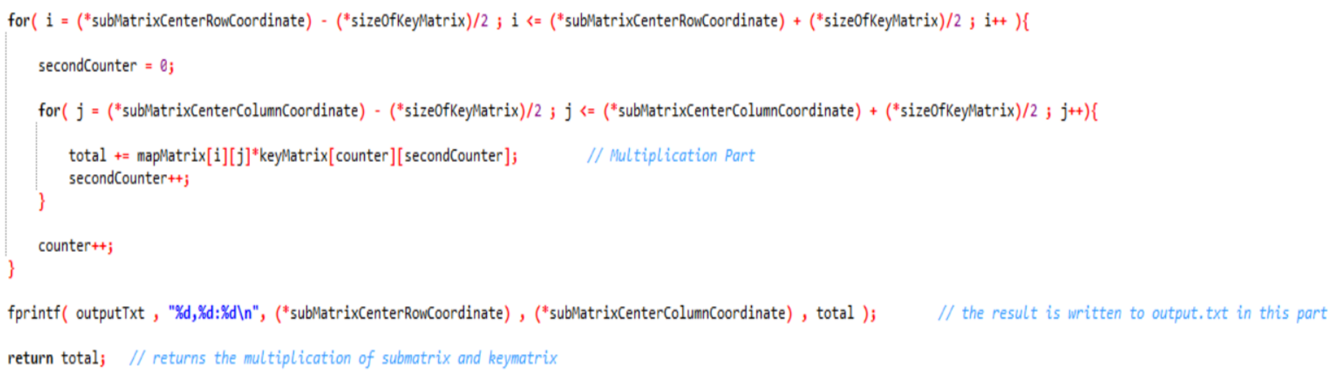
```
int result = mod( outputTxt , mapMatrix , keyMatrix , subMatrixCenterRowCoordinate , subMatrixCenterColumnCoordinate , sizeOfKeyMatrix );
```



```
if( result%5 == 0){

    return;
}
else{

    if       ( result%5 == 1 && ( (*subMatrixCenterRowCoordinate)    - (*sizeOfKeyMatrix) ) < 0                  )     (*subMatrixCenterRowCoordinate)    +=  (*sizeOfKeyMatrix);
    else if  ( result%5 == 1                                                                                      )     (*subMatrixCenterRowCoordinate)    -=  (*sizeOfKeyMatrix);
    else if  ( result%5 == 2 && ( (*subMatrixCenterRowCoordinate)    + (*sizeOfKeyMatrix) ) > *heightOfMapMatrix  )     (*subMatrixCenterRowCoordinate)    -=  (*sizeOfKeyMatrix);
    else if  ( result%5 == 2                                                                                      )     (*subMatrixCenterRowCoordinate)    +=  (*sizeOfKeyMatrix);
    else if  ( result%5 == 3 && ( (*subMatrixCenterColumnCoordinate) + (*sizeOfKeyMatrix) ) > *widthOfMapMatrix   )     (*subMatrixCenterColumnCoordinate) -=  (*sizeOfKeyMatrix);
    else if  ( result%5 == 3                                                                                      )     (*subMatrixCenterColumnCoordinate) +=  (*sizeOfKeyMatrix);
    else if  ( result%5 == 4 && ( (*subMatrixCenterColumnCoordinate) - (*sizeOfKeyMatrix) ) < 0                  )     (*subMatrixCenterColumnCoordinate) +=  (*sizeOfKeyMatrix);
    else if  ( result%5 == 4                                                                                      )     (*subMatrixCenterColumnCoordinate) -=  (*sizeOfKeyMatrix);

    search( outputTxt , mapMatrix , keyMatrix , subMatrixCenterRowCoordinate , subMatrixCenterColumnCoordinate , sizeOfKeyMatrix , heightOfMapMatrix , widthOfMapMatrix);
}
```

❻ In mod.c :

⦿ **This function returns the value that is multiplicaton of keyMatrix and any mapMatrix's subMatrix according to sizeOfKeyMatrix. Also the results are written to output file.**



```
for( i = (*subMatrixCenterRowCoordinate) - (*sizeOfKeyMatrix)/2 ; i <= (*subMatrixCenterRowCoordinate) + (*sizeOfKeyMatrix)/2 ; i++ ){

    secondCounter = 0;

    for( j = (*subMatrixCenterColumnCoordinate) - (*sizeOfKeyMatrix)/2 ; j <= (*subMatrixCenterColumnCoordinate) + (*sizeOfKeyMatrix)/2 ; j++){

        total += mapMatrix[i][j]*keyMatrix[counter][secondCounter];         // Multiplication Part
        secondCounter++;
    }

    counter++;
}

fprintf( outputTxt , "%d,%d:%d\n", (*subMatrixCenterRowCoordinate) , (*subMatrixCenterColumnCoordinate) , total );        // the result is written to output.txt in this part

return total;   // returns the multiplication of submatrix and keymatrix
```

❼ Makefile :

⦿ This file provides just writing "make" to compile the code on the dev system.

```
output: main.o setMapMatrix.o setKeyMatrix.o search.o mod.o
    g++ main.o setMapMatrix.o setKeyMatrix.o search.o mod.o -o findtreasure

main.o: main.c
    g++ -c main.c

setMapMatrix.o: setMapMatrix.c functions.h
    g++ -c setMapMatrix.c

setKeyMatrix.o: setKeyMatrix.c functions.h
    g++ -c setKeyMatrix.c

search.o: search.c functions.h
    g++ -c search.c

mod.o: mod.c functions.h
    g++ -c mod.c

clean:
    rm *.o findtreasure
```