Bairam Kuliev Computer Engineering

36180633

In this experiment, we have to find a treasure on a treasure map designed as a matrix. For resolving this problem we have a map matrix and key matrix. All the information about map matrix and key matrix given in file, so we have to read all of them from file. The result of the multiplication of the map matrix and key matrix must give in which direction to go on the treasure map.

An example of treasure map matrix.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	1	5	8	6	9	4	7	5	1	2	9	2	8	3	6	0
0	1	5	8	3	2	8	4	6	9	3	7	4	6	8	9	3	0
0	1	6	2	9	3	5	8	9	2	6	1	8	9	7	3	1	0
0	5	7	5	8	1	3	4	8	5	1	5	8	9	3	5	2	0
0	9	4	7	5	1	2	9	4	7	5	1	2	9	2	8	3	0
0	8	4	6	9	3	7	8	4	6	9	3	7	4	6	8	9	0
0	5	8	9	2	6	1	5	8	9	2	6	1	8	9	7	3	0
0	3	8	1	3	4	8	5	1	5	3	8	1	3	4	8	5	0
0	7	5	1	2	9	4	7	5	1	7	5	1	2	9	4	7	0
0	6	9	3	7	8	4	6	9	3	6	9	3	7	8	4	6	0
0	9	2	6	1	5	8	9	2	3	4	8	5	1	5	8	9	0
0	8	4	6	9	3	7	3	7	2	9	4	7	5	1	2	9	0
0	5	8	9	2	6	1	6	1	7	8	4	6	9	3	7	4	0
0	3	8	1	3	4	8	2	9	1	5	8	9	2	6	1	8	0
0	7	5	1	2	9	4	8	1	3	4	8	8	4	6	8	4	0
0	8	4	6	9	3	7	5	1	2	9	4	5	8	9	5	8	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

An example of key matrix.

0	-1	0
-1	20	-1
0	-1	0

We can change our position in 4 directions:

1.Go up

2.Go down

3.Go right

4.Go left

And 0 if we will find treasure

We always start from the top left corner of the map matrix.

Start search:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 5 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 \\ -1 & 20 & -1 \\ 0 & -1 & 0 \end{bmatrix} = 58$$

After that we have to we have to take the remainder of the division by 5 and it give us the second direction.

58 % 5 = 3

If we can not go to the this direction because of out of map matrix we have to change our direction in the opposite direction. If the mod result is 0 so it means that we found the treasure. All output have to be written in the file output.txt in this format

```
1,1:58
1,4:146
4,4:140
```

There each line contain the midpoint of the sub-matrix and also the result of multiplication.

For solving this problem I will use dynamic memory allocation, recursion, two dimensional array.

Let start with introduce my values.

```
int **mapptr,**keyptr;
int i, j, row, column,sizekey=atoi(argv[2]);
char *token;
FILE *map;
FILE *key;
FILE *output;
```

I have two multidimensional array **mapptr and **keyptr where I will allocate memory for keeping treasure map and key matrices respectively. Then I have integer row, column and sizekey which will help me to create a loop for reading map and key matrices from the .txt files to the my two-dimensional arrays. I also have integer numbers i and j, which will help me keep the position of each number on the map and key matrices. Then I make character pointer *token, which I will use in the atoi function to create an integers from character. Then I define File pointers, *map and *key I will use to read from the file and scan them to my multidimensional array **mapptr and **keyptr, *output I will use for print there the output of multiplication map and key matrices.

I will receive information from the command line about size of map and key matrices, also the name of three .txt file, so because of that I will use this format `void main(int argc, char *argv[])`

where int argc is number of commands and char *argv[] the array which will keep all of commands.

All the commands which will be given is character format, but I have information about size of map and key matrices which is integer, so for converting characters to integers I will use atoi and strtok functions.

Atoi will help me to convert character to integer, strtok will help to take two integer information about row and column size of map matrix. Firstly, this function take the first part from some number to the x and then stop, it will be row of map matrix, then I do this for column of map matrix, but in this time it start from x to the NULL.

```
token = strtok(argv[1],"x");
row = atoi(token);
token = strtok(NULL, "x");
column = atoi(token);
```

Then I allocate memory for map and key matrices with malloc. Firstly, I allocate memory to keep all of given rows, then I create loop for keeping the information about every number in this row respectively, it will my column, then I do the same thing for key matrix.

```
  // memory allocation for map matrix
mapptr = (int**)malloc(row * sizeof(int*));
for (i = 0; i<row; i++)
  mapptr[i] = (int*)malloc(column * sizeof(int));

printf("\n");

  // memory allocation for key matrix
keyptr = (int**)malloc(sizekey * sizeof(int*));
for (i = 0; i<sizekey; i++)
  keyptr[i] = (int*)malloc(sizekey * sizeof(int));
```

After that I open given files, map and key files I will read, but output I will use for writing all outputs of multiplication. For this I will use fopen function.

```
map=fopen(argv[3],"r");
key=fopen(argv[4],"r");
output=fopen(argv[5],"w");
```

Then I am going to read map and key files, and write all the elements from the file to the **mapptr and **keyptr

arrays. Firstly, I made the loop for row, and then in this loop I made an another loop for column, in this time function fscanf will help me to read all information from file and write them into the arrays. For key matrix I make something similar, but we always have a square matrix, so I can do two loop between i and sizekey, and between j and sizekey.

```
//reading map matrix elements from file
for (i = 0; i < row; i++)  // loop for row
{
  for (j = 0; j < column; j++)  // loop for column
    fscanf(map,"%d ",&mapptr[i][j]);
}
//reading key matrix elements from file
for (i = 0; i < sizekey; i++)// loop for row
{
  for (j = 0; j < sizekey; j++)
    fscanf(key,"%d ",&keyptr[i][j]);// loop for column
}
```

Then I decided to show all the values on the display, to be sure, for this I use printf function and two loops.

```
//loop for writing map matrix elements on screen
printf("Map matrix: \n");
for (i = 0; i<row; i++)
{
  for (j = 0; j<column; j++)
  {
      printf("%d  ", mapptr[i][j]);
  }
      printf("\n");
}

printf("\n");
//loop for writing key matrix elements on screen
printf("Key matrix: \n");
for (i = 0; i<sizekey; i++)
{
  for (j = 0; j<sizekey; j++)
  {
      printf("%d  ", keyptr[i][j]);
  }
      printf("\n");
}
```

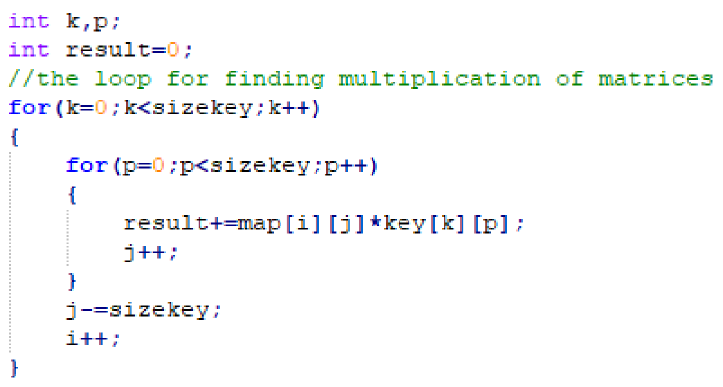
After that I just close map and key files, for this I use fclose function.

```
fclose(map);
fclose(key);
```

Then I call my findtreasure function, firstly before that, I give to i and j 0, because i and j the index of every number in the map matrix, and it always have to start from the top left corner of the map matrix. When I call my function I send information about output file, where I will keep all of outputs, size of key matrix, the information about row and column numbers, address of map matrix, where is the keeping all of my treasure map matrix, key pointer where also key matrix, i and j indexes.

```
i=0;j=0;
findtreasure(output, sizekey, row, column, mapptr, keyptr, i, j);
```

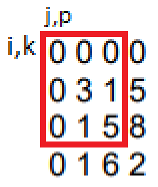
The i and j always will follow position on the map, so they will change always, so it is important part in my recursion. Then, I make multiplication the first part of map and key, for this I take just square which equal to the size of key, I make two loops, first for row and second for column. The index of key matrix is always static, which means that it will be start always from 0 to the size of key matrix. For example, if sizekey=3, so k and p will always between 0 and 3. It is give me the best method to multiplication, because my loop will always between 0 and the numbers of seize of key, and at this time i and j will always change, that will give me every time the new position on the map matrix.



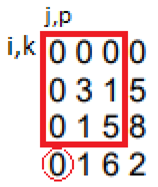
```
int k,p;
int result=0;
//the loop for finding multiplication of matrices
for(k=0;k<sizekey;k++)
{
    for(p=0;p<sizekey;p++)
    {
        result+=map[i][j]*key[k][p];
        j++;
    }
    j-=sizekey;
    i++;
}
```

At first time all the integers i,k,j,p is 0, but then i and j will keep going and give information about current position.

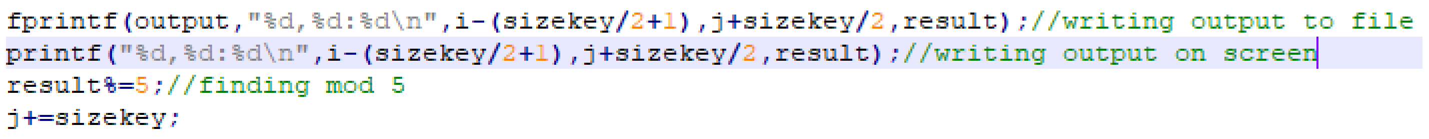
For example, if our size of key 3, so j have to go 3 times to the left, which means sizekey also, it will happen in the second loop, and then when the loop will finish I again give the j his first value, so I subtract size of key, and that again until k<sizekey.



So after all operations I have the information about position of my current location and result of multiplication. Now our i=4 and j=0 for this example

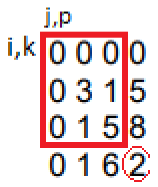


After all operations I am going write my outputs to the output file for this I use fprintf function. In the output I have to give the information about midpoint of sub-matrix, we know that sizekey will always odd number like 3,5,7 etc, and so for this I make a formula, for i, i = i - sizekey/2+1,so before we had i=3,but now i=1, and also formula for j, j = j + sizekey/2,and now our j=1.It will always change depending on sizekey, I and j, but always will give the midpoint our sub-matrix. Then I print otput on screen, just to be sure, then I take remainder of the division 5, after that I give j+=sizekey because of it give the good position to going to the next step.

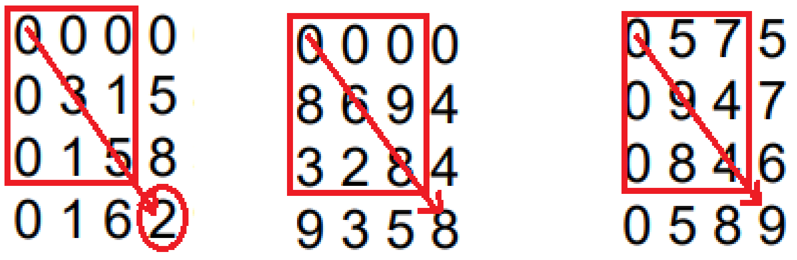


```
fprintf(output,"%d,%d:%d\n",i-(sizekey/2+1),j+sizekey/2,result);//writing output to file
printf("%d,%d:%d\n",i-(sizekey/2+1),j+sizekey/2,result);//writing output on screen
result%=5;//finding mod 5
j+=sizekey;
```

And now I am here



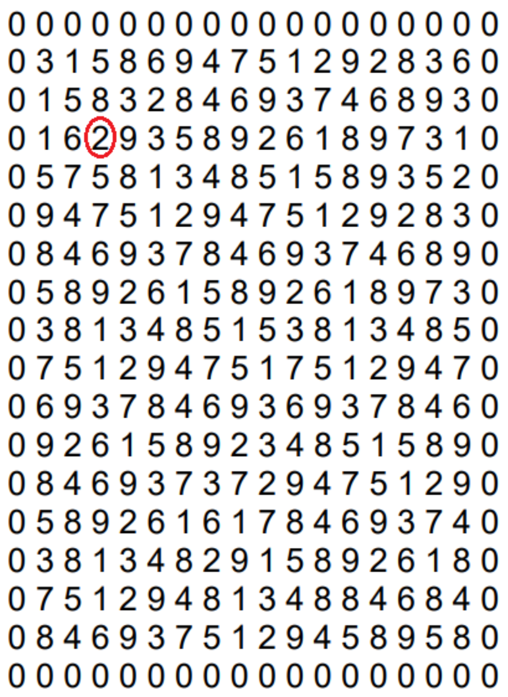
I understand that I always, after the doing processes above, will have an interesting position, and it will happen always regardless of size of key, when the inputs of map and key matrices will change, it means I will have position of i and j size of key+1, for example if sizekey=3, i=4 and j=4, if sizekey=5, i=6 and j=6.



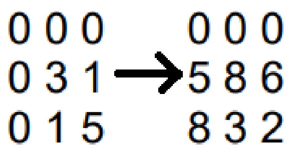
Then start the important part of this program for solution our problem.

If result not 0 keep going.

Now our position is here and result is 3, so we have to go to the right.



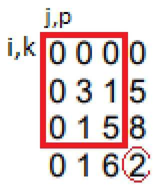
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 1 5 8 6 9 4 7 5 1 2 9 2 8 3 6 0
0 1 5 8 3 2 8 4 6 9 3 7 4 6 8 9 3 0
0 1 6 2 9 3 5 8 9 2 6 1 8 9 7 3 1 0
0 5 7 5 8 1 3 4 8 5 1 5 8 9 3 5 2 0
0 9 4 7 5 1 2 9 4 7 5 1 2 9 2 8 3 0
0 8 4 6 9 3 7 8 4 6 9 3 7 4 6 8 9 0
0 5 8 9 2 6 1 5 8 9 2 6 1 8 9 7 3 0
0 3 8 1 3 4 8 5 1 5 3 8 1 3 4 8 5 0
0 7 5 1 2 9 4 7 5 1 7 5 1 2 9 4 7 0
0 6 9 3 7 8 4 6 9 3 6 9 3 7 8 4 6 0
0 9 2 6 1 5 8 9 2 3 4 8 5 1 5 8 9 0
0 8 4 6 9 3 7 3 7 2 9 4 7 5 1 2 9 0
0 5 8 9 2 6 1 6 1 7 8 4 6 9 3 7 4 0
0 3 8 1 3 4 8 2 9 1 5 8 9 2 6 1 8 0
0 7 5 1 2 9 4 8 1 3 4 8 8 4 6 8 4 0
0 8 4 6 9 3 7 5 1 2 9 4 5 8 9 5 8 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



$$\begin{matrix}0&0&0\\0&3&1\\0&1&5\end{matrix} \rightarrow \begin{matrix}0&0&0\\5&8&6\\8&3&2\end{matrix}$$

For this my index of i and j must be change from i=4 and j=4 to i=0 and j=4, so there just change the value of i, so i-=sizekey.

Let's suppose other situations, for example when in this position we have another way.



$$\begin{array}{c|cccc} & j,p & & & \\ \hline i,k & 0 & 0 & 0 & 0 \\ & 0 & 3 & 1 & 5 \\ & 0 & 1 & 5 & 8 \\ & 0 & 1 & 6 & 2 \end{array}$$

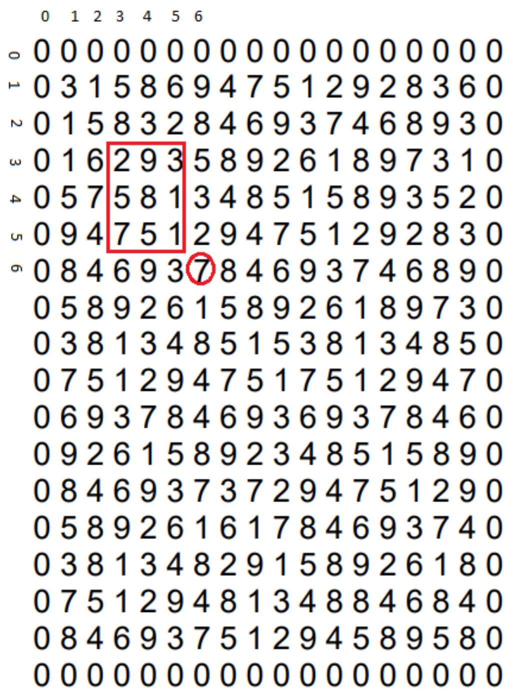
For example, our result is 4, so we have to go to the left, but we now that we can not do this, because of the boundary of map, so we have to go to the right, for this we have condition if(j- sizekey == 0) it means I can not go to the left, so I will not change the value of j, I just change the value of i, i-=sizekey.

But what if we can go to the left, for example in this situation, so now i and j, both of them must change, i will be 3 and j will be 0, for this I can write condition

```
if(j-sizekey!=0)
        j - =sizekey*2;
    i - = sizekey;
```

Suppose in this duration our result is 1, so i and j must be change, i will be 0 and j will be 3, for this situation I can write condition

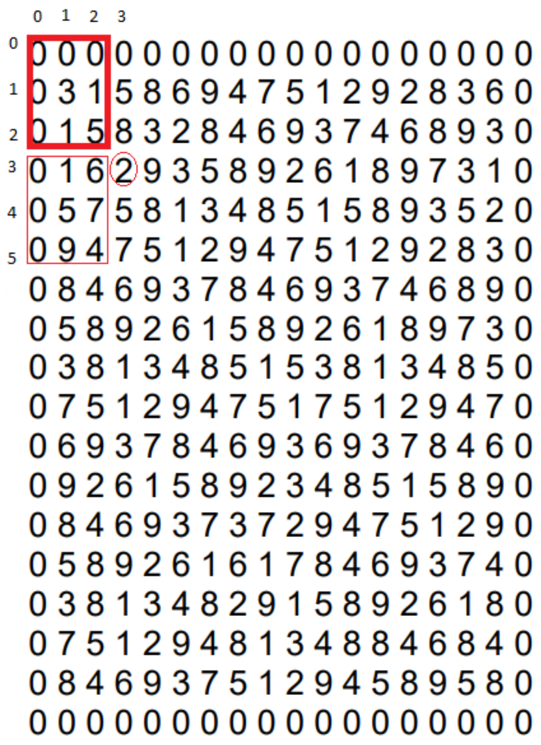
```
if(i - sizekey!=0)
        i - = sizekey*2;
      j - = sizekey;
```



	0	1	2	3	4	5	6											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3	1	5	8	6	9	4	7	5	1	2	9	2	8	3	6	0
2	0	1	5	8	3	2	8	4	6	9	3	7	4	6	8	9	3	0
3	0	1	6	2	9	3	5	8	9	2	6	1	8	9	7	3	1	0
4	0	5	7	5	8	1	3	4	8	5	1	5	8	9	3	5	2	0
5	0	9	4	7	5	1	2	9	4	7	5	1	2	9	2	8	3	0
6	0	8	4	6	9	3	7	8	4	6	9	3	7	4	6	8	9	0
	0	5	8	9	2	6	1	5	8	9	2	6	1	8	9	7	3	0
	0	3	8	1	3	4	8	5	1	5	3	8	1	3	4	8	5	0
	0	7	5	1	2	9	4	7	5	1	7	5	1	2	9	4	7	0
	0	6	9	3	7	8	4	6	9	3	6	9	3	7	8	4	6	0
	0	9	2	6	1	5	8	9	2	3	4	8	5	1	5	8	9	0
	0	8	4	6	9	3	7	3	7	2	9	4	7	5	1	2	9	0
	0	5	8	9	2	6	1	6	1	7	8	4	6	9	3	7	4	0
	0	3	8	1	3	4	8	2	9	1	5	8	9	2	6	1	8	0
	0	7	5	1	2	9	4	8	1	3	4	8	8	4	6	8	4	0
	0	8	4	6	9	3	7	5	1	2	9	4	5	8	9	5	8	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Now I suppose, if our result is 2 we have to go down, so for this situation I must just change the value of j, because i already right, so j - = sizekey.

Then suppose that in this position our result is 1, so I have to go to the up, but we now that we can not do this, because of the boundary of map, so I have to go to the opposite side, in this situation if(i - sizekey == 0) it means that I can not go to the up, so I must change just value of j, in order to go to the down j - = sizekey.



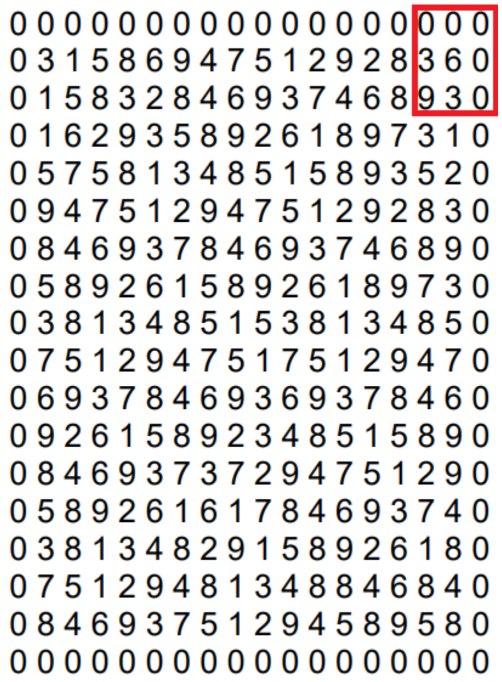
	0	1	2	3														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3	1	5	8	6	9	4	7	5	1	2	9	2	8	3	6	0
2	0	1	5	8	3	2	8	4	6	9	3	7	4	6	8	9	3	0
3	0	1	6	2	9	3	5	8	9	2	6	1	8	9	7	3	1	0
4	0	5	7	5	8	1	3	4	8	5	1	5	8	9	3	5	2	0
5	0	9	4	7	5	1	2	9	4	7	5	1	2	9	2	8	3	0
	0	8	4	6	9	3	7	8	4	6	9	3	7	4	6	8	9	0
	0	5	8	9	2	6	1	5	8	9	2	6	1	8	9	7	3	0
	0	3	8	1	3	4	8	5	1	5	3	8	1	3	4	8	5	0
	0	7	5	1	2	9	4	7	5	1	7	5	1	2	9	4	7	0
	0	6	9	3	7	8	4	6	9	3	6	9	3	7	8	4	6	0
	0	9	2	6	1	5	8	9	2	3	4	8	5	1	5	8	9	0
	0	8	4	6	9	3	7	3	7	2	9	4	7	5	1	2	9	0
	0	5	8	9	2	6	1	6	1	7	8	4	6	9	3	7	4	0
	0	3	8	1	3	4	8	2	9	1	5	8	9	2	6	1	8	0
	0	7	5	1	2	9	4	8	1	3	4	8	8	4	6	8	4	0
	0	8	4	6	9	3	7	5	1	2	9	4	5	8	9	5	8	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Assume that we are in this duration and the result = 3, so i=3 and j=18, but we now that we can not do this, because of the boundary of map, so I have to go to the left, so now we have condition if (j == column) in this situation we change both of index, i and j, so I can write this condition like this

```
if( j == column)

        j - = sizekey*2;

      I - = sizekey;
```



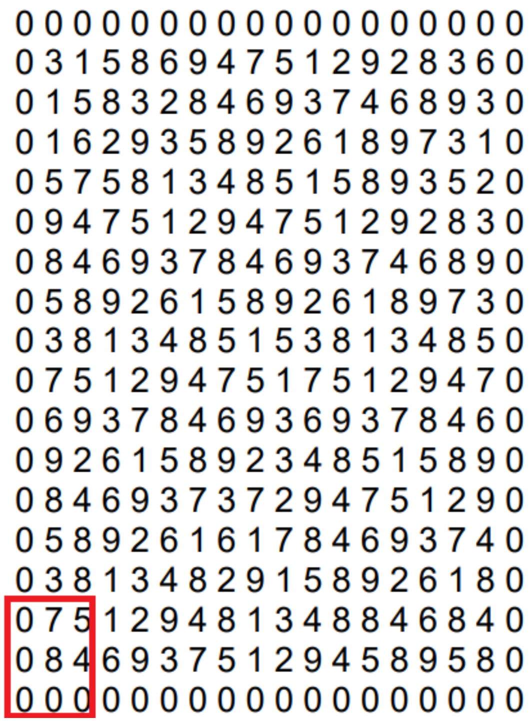
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 1 5 8 6 9 4 7 5 1 2 9 2 8 3 6 0
0 1 5 8 3 2 8 4 6 9 3 7 4 6 8 9 3 0
0 1 6 2 9 3 5 8 9 2 6 1 8 9 7 3 1 0
0 5 7 5 8 1 3 4 8 5 1 5 8 9 3 5 2 0
0 9 4 7 5 1 2 9 4 7 5 1 2 9 2 8 3 0
0 8 4 6 9 3 7 8 4 6 9 3 7 4 6 8 9 0
0 5 8 9 2 6 1 5 8 9 2 6 1 8 9 7 3 0
0 3 8 1 3 4 8 5 1 5 3 8 1 3 4 8 5 0
0 7 5 1 2 9 4 7 5 1 7 5 1 2 9 4 7 0
0 6 9 3 7 8 4 6 9 3 6 9 3 7 8 4 6 0
0 9 2 6 1 5 8 9 2 3 4 8 5 1 5 8 9 0
0 8 4 6 9 3 7 3 7 2 9 4 7 5 1 2 9 0
0 5 8 9 2 6 1 6 1 7 8 4 6 9 3 7 4 0
0 3 8 1 3 4 8 2 9 1 5 8 9 2 6 1 8 0
0 7 5 1 2 9 4 8 1 3 4 8 8 4 6 8 4 0
0 8 4 6 9 3 7 5 1 2 9 4 5 8 9 5 8 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Let suppose if my location when i=18 and j=3, and result is 2, so now we can not go to the down side, because of the boundary of map, so I have to go to the up, for this situation we have condition if(i==row), it is mean I can not go to the down and we have to change value of i, so i must be 12 and j=0, so I can write like this

```
if(i==row)

        i-=sizekey*2;

      j-=sizekey;
```



```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 1 5 8 6 9 4 7 5 1 2 9 2 8 3 6 0
0 1 5 8 3 2 8 4 6 9 3 7 4 6 8 9 3 0
0 1 6 2 9 3 5 8 9 2 6 1 8 9 7 3 1 0
0 5 7 5 8 1 3 4 8 5 1 5 8 9 3 5 2 0
0 9 4 7 5 1 2 9 4 7 5 1 2 9 2 8 3 0
0 8 4 6 9 3 7 8 4 6 9 3 7 4 6 8 9 0
0 5 8 9 2 6 1 5 8 9 2 6 1 8 9 7 3 0
0 3 8 1 3 4 8 5 1 5 3 8 1 3 4 8 5 0
0 7 5 1 2 9 4 7 5 1 7 5 1 2 9 4 7 0
0 6 9 3 7 8 4 6 9 3 6 9 3 7 8 4 6 0
0 9 2 6 1 5 8 9 2 3 4 8 5 1 5 8 9 0
0 8 4 6 9 3 7 3 7 2 9 4 7 5 1 2 9 0
0 5 8 9 2 6 1 6 1 7 8 4 6 9 3 7 4 0
0 3 8 1 3 4 8 2 9 1 5 8 9 2 6 1 8 0
0 7 5 1 2 9 4 8 1 3 4 8 8 4 6 8 4 0
0 8 4 6 9 3 7 5 1 2 9 4 5 8 9 5 8 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

After all this part I recall my function by using the recursion, all the information about output, sizekey, row, column, map and key are the same, but i and j will be change, that give us the next location of sub matrix.

```
findtreasure(output, sizekey, row, column, map,key, i, j);//recursive function
```

If our result is 0 , so we can print that we found treasure and close output file.

Then I have to free all the allocated memory, for this purposes I use the loops and free function.

```
  //free memory allocation of map matrix
for (i = 0; i < row; i++)
  free(mapptr[i]);
free(mapptr);
  //free memory allocation of key matrix
for (i = 0; i < sizekey; i++)
  free(keyptr[i]);
free(keyptr);
```

In the program I was finking to use structure, for keep all the information in one place, but I did not find it is clearly to understand for me every step better then just can I write all the information step by step like this, so I did not use it.

```
findtreasure(output, sizekey, row, column, map,key, i, j);//recursive function
```