DATA STRUCTURES LABORATORY

Assignment 2

Name: **Katkar Prathamesh Shivaji**

Enrollment No.: 18114038

Batch : O2

Submission Date: 07 August 2019

Branch: CSE

Email ID:

1. kshivaji@cs.iitr.ac.in (Piazza)

2. prathameshkatkar11@gmail.com (Moodle)

Problem Statement 1:

In this Problem, you have to implement a simple transposition cipher, where this cipher encrypts and decrypts a sequence of characters by dividing the sequence into blocks of size n, where n is specified by the encryption key. If the input text has a length that is not a multiple of n, the last block is padded with null characters ('\0'). In addition to n, the key also specifies two parameters a and b. For each block, the i-th output character, starting from 0 as usual, is set to the j-th input character, where j = (ai + b) mod n. For appropriate choices of a and b, this will reorder the characters in the block in a way that can be reversed by choosing a corresponding decryption key (n, a′, b′). For example, if n = 5, a = 3, and b = 2, the string Hello, world! would be encrypted like this:

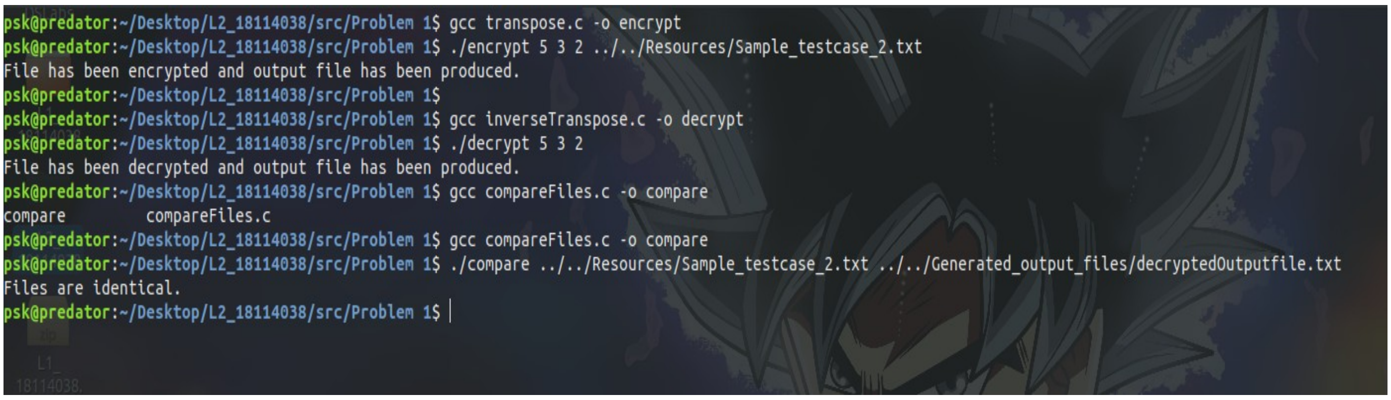
in:	H	e	l	l	o	,		w	o	r	l	d	!	\0	\0
i:	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
j:	2	0	3	1	4	2	0	3	1	4	2	0	3	1	4
out:	l	H	l	e	o	w	,	o		r	!	l	\0	d	\o

Task to Perform Write a program transpose.c that takes n, a, b, inputfile.txt in argv[1], argv[2], argv[3], and argv[4], respectively, applies the above encryption; and writes the result to outputfile.txt. Further, write a program inverseTranspose.c that decrypt the outputfile.txt and result in a new file named decryptedOutputfile.txt. Finally, write a program compareFiles.c to find the equivalence between the inputfile.txt and decryptedOutputfile.txt files. You may assume that n, a, and b are all small enough to fit into variables of type int. Your program should exit with a nonzero exit code if n is not at least 1 or if it is not given exactly four arguments, but you do not need to do anything to test for badly-formatted arguments. You should not make any other assumptions about the values of n, a, or b; for example, either of a or b could be zero or negative.

Data Structures used:

1. Arrays were used to hold the data in the form of character arrays.

Snapshots of code running:



```
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ gcc transpose.c -o encrypt
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ ./encrypt 5 3 2 ../../Resources/Sample_testcase_2.txt
File has been encrypted and output file has been produced.
psk@predator:~/Desktop/L2_18114038/src/Problem 1$
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ gcc inverseTranspose.c -o decrypt
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ ./decrypt 5 3 2
File has been decrypted and output file has been produced.
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ gcc compareFiles.c -o compare
compare         compareFiles.c
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ gcc compareFiles.c -o compare
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ ./compare ../../Resources/Sample_testcase_2.txt ../../Generated_output_files/decryptedOutputfile.txt
Files are identical.
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ 
```

Reported Execution time:

1. transpose.c

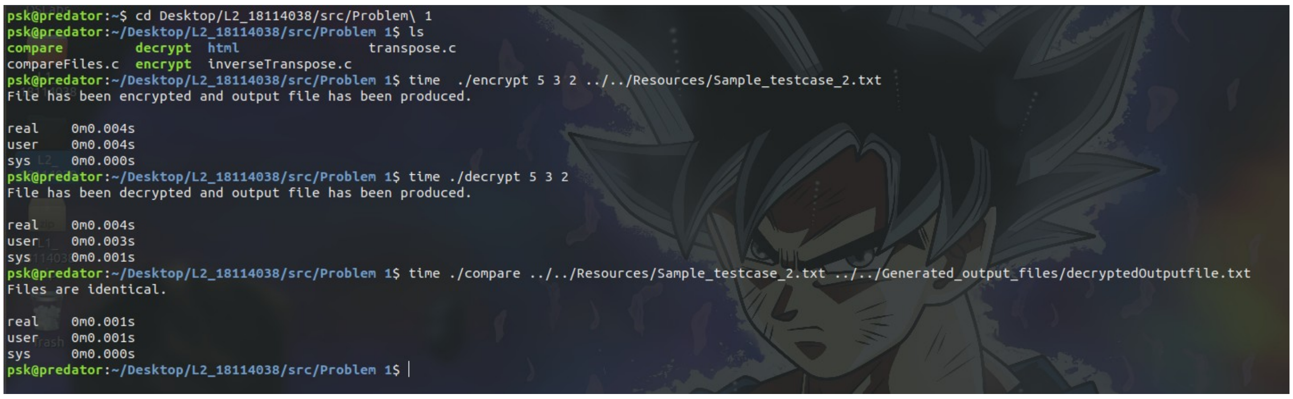
```
real    0m0.004s
user    0m0.004s
sys     0m0.000s
```

2. inverseTranspose.c

```
real    0m0.004s
user    0m0.003s
sys     0m0.001s
```

3. compareFiles.c

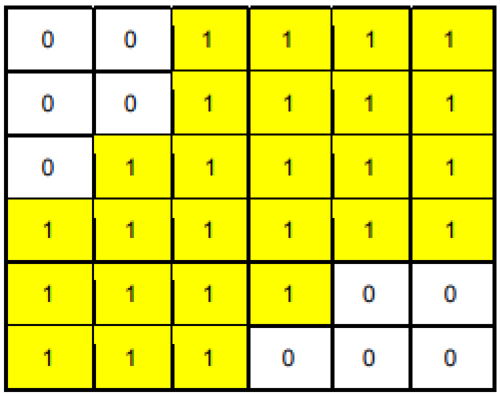
```
real    0m0.001s
user    0m0.001s
sys     0m0.000s
```



```
psk@predator:~$ cd Desktop/L2_18114038/src/Problem\ 1
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ ls
compare         decrypt  html                 transpose.c
compareFiles.c  encrypt  inverseTranspose.c
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ time  ./encrypt 5 3 2 ../../Resources/Sample_testcase_2.txt
File has been encrypted and output file has been produced.

real    0m0.004s
user    0m0.004s
sys     0m0.000s
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ time ./decrypt 5 3 2
File has been decrypted and output file has been produced.

real    0m0.004s
user    0m0.003s
sys     0m0.001s
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ time ./compare ../../Resources/Sample_testcase_2.txt ../../Generated_output_files/decryptedOutputfile.txt
Files are identical.

real    0m0.001s
user    0m0.001s
sys     0m0.000s
psk@predator:~/Desktop/L2_18114038/src/Problem 1$ |
```

Problem Statement 2:

A region can be represented either by its interior or by its boundary. Here we represent the region by its interior using one of the most common methods called image array. In this case we have a collection of pixels. Since the number of elements in the array can be quite large, the main objective is to reduce its size by aggregating equal-valued pixels.



0	0	1	1	1	1
0	0	1	1	1	1
0	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	0	0
1	1	1	0	0	0

Fig 1: Image array representation of a region

A general approach is to treat the region as a quadtree, where the region is represented as a union of maximal non-overlapping square blocks whose sides are in power of 2. The quadtree can be generated by successive subdivision of the image array into four equal sized quadrants. If the sub-array does not consist entirely of 1s or entirely of 0s, it is then further subdivided into quadrants and sub-quadrants, etc.

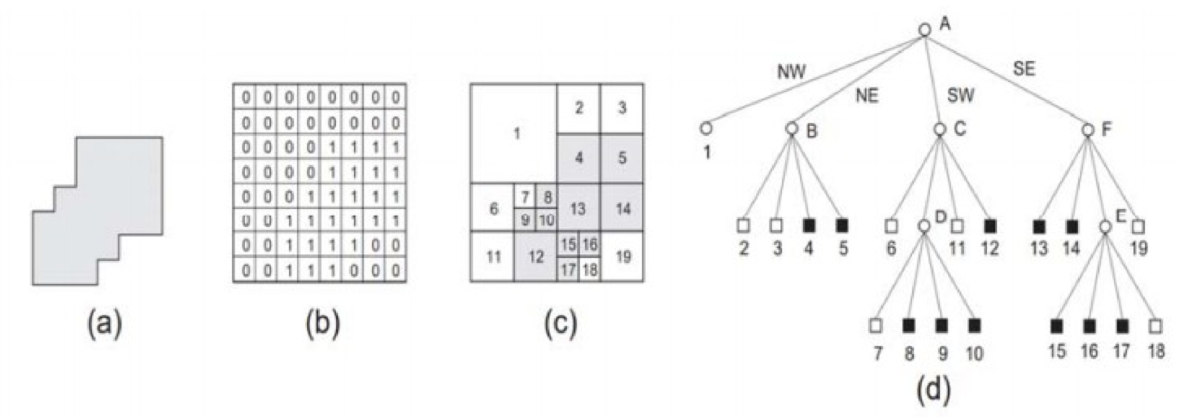


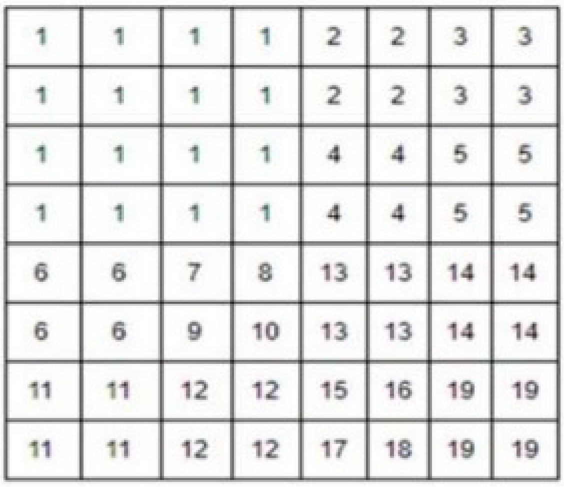
Fig 2: Flow of quadtree representation from the sample region

Example: The above figures represent a region and its corresponding quadtree representation, (a) Defined as the Sample region having all the bit value to 1. (b) Defined as the binary array of size 8*8 which having bit value other than sample region within it is 0. (c) Represent the conversion of binary array into the blocks where each block is formed as a union of maximal square having the same bit value. This kind of array is called a maximal square array. (d) Quadtree representation, where the root node corresponds to the entire array. Each son of a node represents a quadrant of the region represented by that node. The leaf nodes of the tree correspond to those blocks for which no further subdivision is necessary. A leaf node is said to be black or white, depending on whether its corresponding block is entirely inside or entirely outside of the represented region. All non-leaf nodes are said to be gray. Task to perform: Write a C program, MAT.c to represent any region (in image array representation), into its quadtree form. Input:

Sample region is represented as n x n array (as shown in Fig. 1 using 6 x 6 matrix). The format of the input file should be as follows: the pixel values in the input file are separated by a single space and rows are separated by a newline character (refer to the sample L2_P2_inputsample.txt file shared in Piazza). (Note: The 6x6 region array should be mapped at the bottom-left corner of a 8x8 binary array as shown in Fig. 2(b))

Output:

1. Print the Maximal square array where it should be filled in the following order: top-right, top-left, bottom-right and bottom-left quadrant, this should be done recursively for all the sub-quadrants. All the cells within a maximal square block should be filled with its corresponding block number. For example, with respect to Fig. 2(c) maximal array should be represented as



1	1	1	1	2	2	3	3
1	1	1	1	2	2	3	3
1	1	1	1	4	4	5	5
1	1	1	1	4	4	5	5
6	6	7	8	13	13	14	14
6	6	9	10	13	13	14	14
11	11	12	12	15	16	19	19
11	11	12	12	17	18	19	19

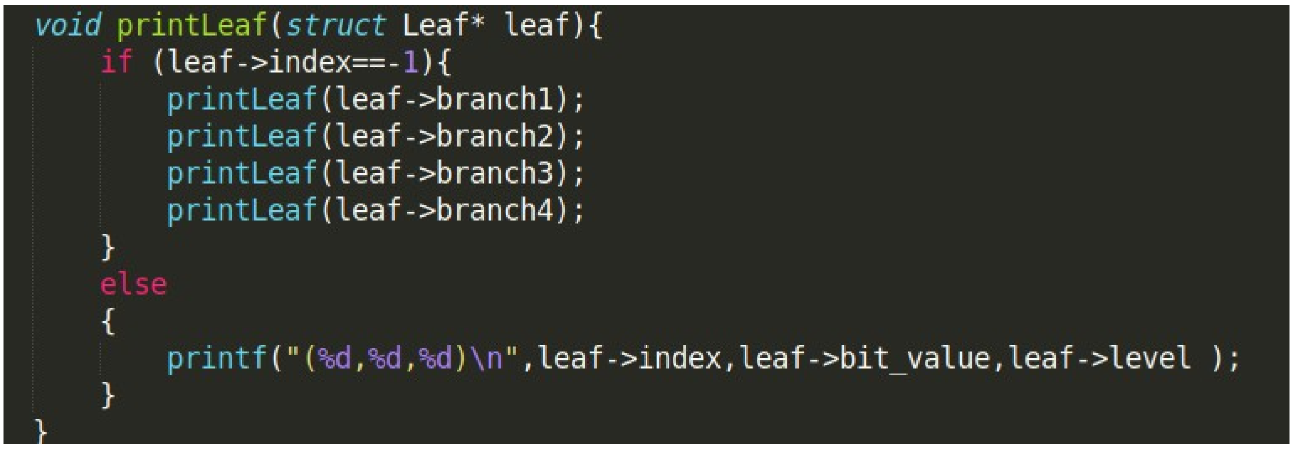
2. Print the quadtree in the following manner, labels of leaf nodes, corresponding bit value and their level information (assuming the level of the root node to be 0), while traversing the quadtree in postorder. For example, in Fig. 2(d) the leaf node 3 having bit value 0 at level 2 and should be printed as (3,0,2).

Data Structures:

1. Have constructed a Quadtree having individual nodes called Leaf

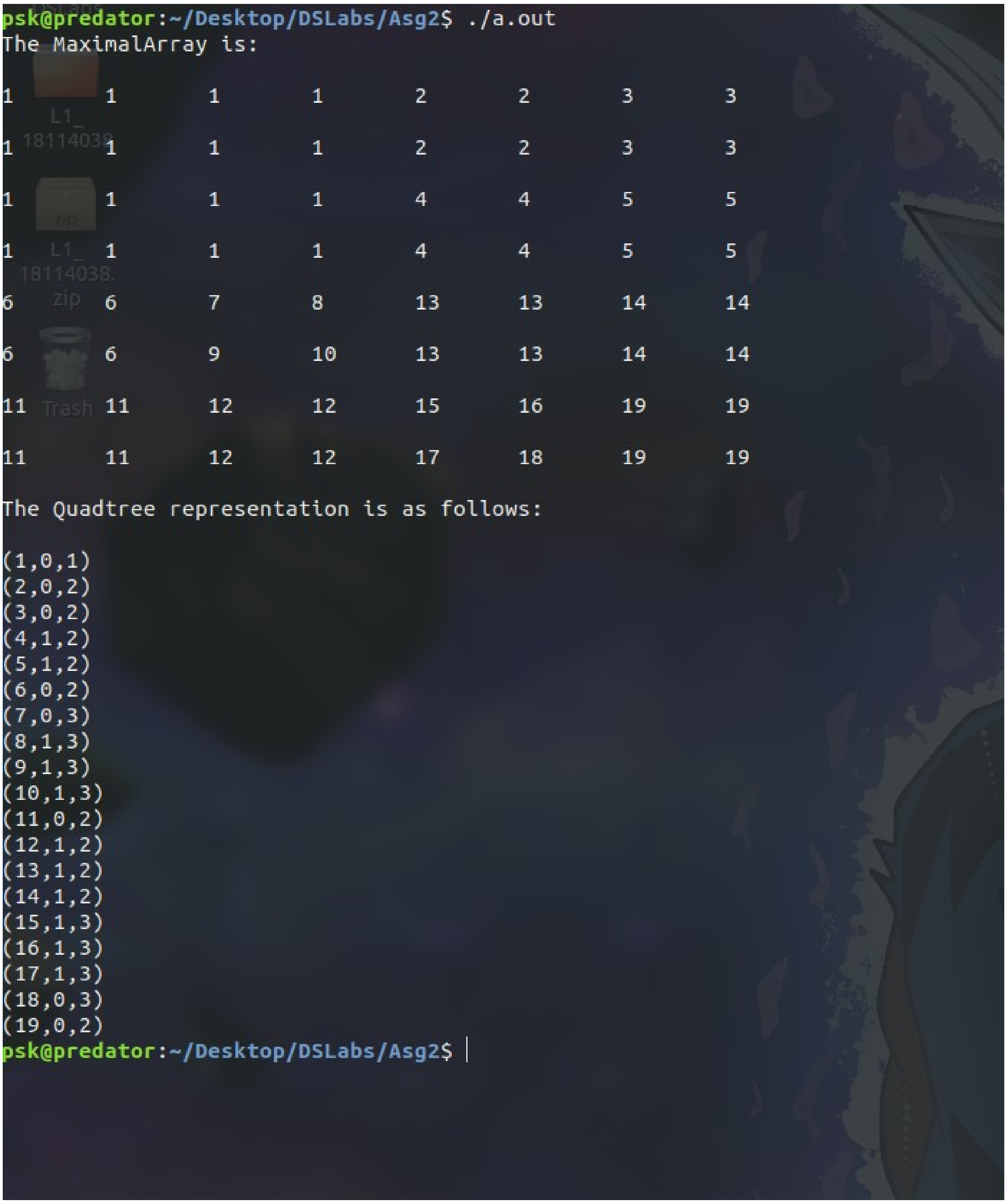
Algorithms used:

Recursive methods for printing the structure of the Quadtree



```c
void printLeaf(struct Leaf* leaf){
    if (leaf->index==-1){
        printLeaf(leaf->branch1);
        printLeaf(leaf->branch2);
        printLeaf(leaf->branch3);
        printLeaf(leaf->branch4);
    }
    else
    {
        printf("(%d,%d,%d)\n",leaf->index,leaf->bit_value,leaf->level );
    }
}
```

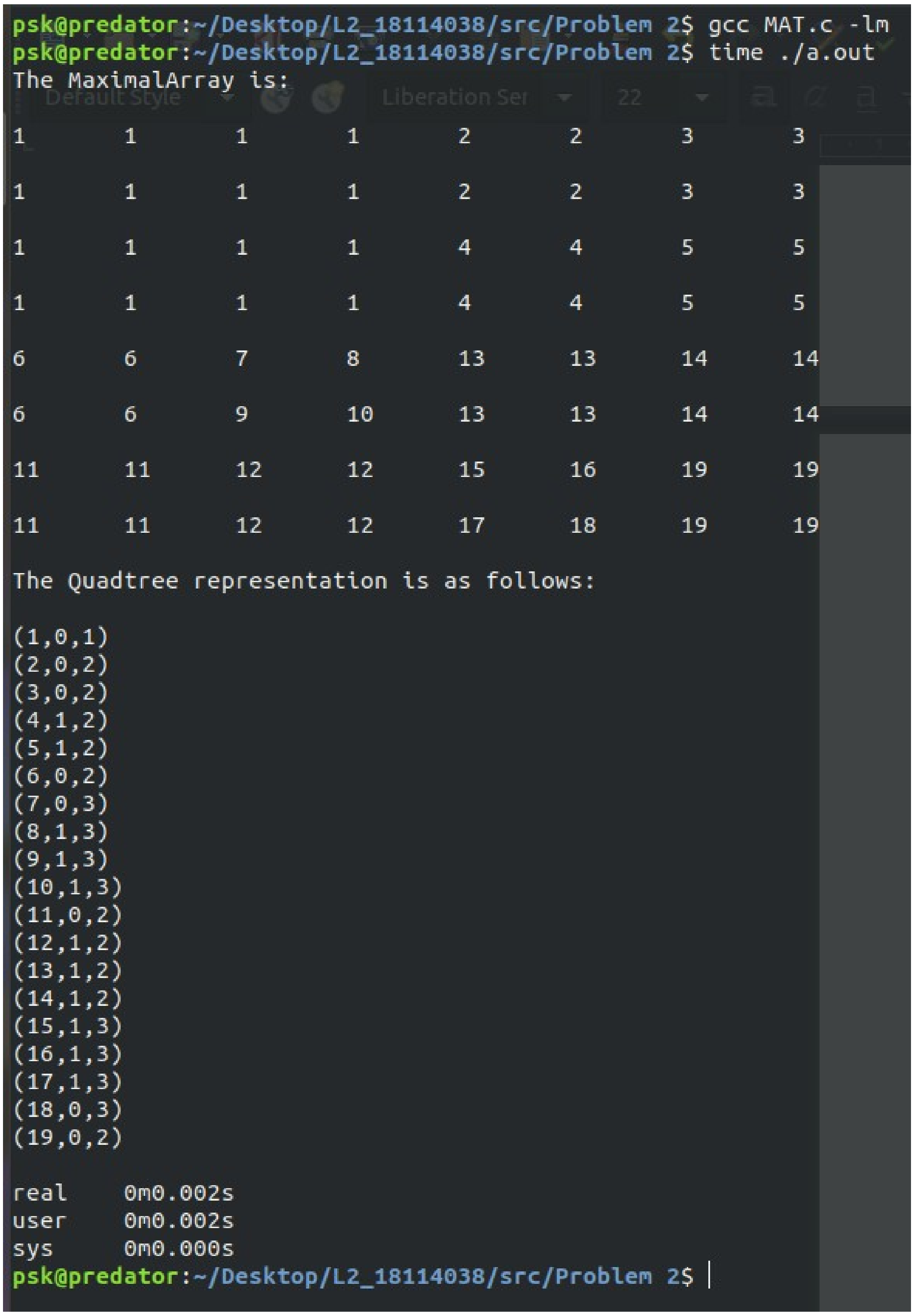
Snapshots of code running:



```
psk@predator:~/Desktop/DSLabs/Asg2$ ./a.out
The MaximalArray is:

1       1       1       1       2       2       3       3

1       1       1       1       2       2       3       3

1       1       1       1       4       4       5       5

1       1       1       1       4       4       5       5

6       6       7       8       13      13      14      14

6       6       9       10      13      13      14      14

11      11      12      12      15      16      19      19

11      11      12      12      17      18      19      19

The Quadtree representation is as follows:

(1,0,1)
(2,0,2)
(3,0,2)
(4,1,2)
(5,1,2)
(6,0,2)
(7,0,3)
(8,1,3)
(9,1,3)
(10,1,3)
(11,0,2)
(12,1,2)
(13,1,2)
(14,1,2)
(15,1,3)
(16,1,3)
(17,1,3)
(18,0,3)
(19,0,2)
psk@predator:~/Desktop/DSLabs/Asg2$
```

Reported Execution Time:

```
real	0m0.002s
user	0m0.002s
sys	0m0.000s
```



```
psk@predator:~/Desktop/L2_18114038/src/Problem 2$ gcc MAT.c -lm
psk@predator:~/Desktop/L2_18114038/src/Problem 2$ time ./a.out
The MaximalArray is:

1       1       1       1       2       2       3       3

1       1       1       1       2       2       3       3

1       1       1       1       4       4       5       5

1       1       1       1       4       4       5       5

6       6       7       8       13      13      14      14

6       6       9       10      13      13      14      14

11      11      12      12      15      16      19      19

11      11      12      12      17      18      19      19

The Quadtree representation is as follows:

(1,0,1)
(2,0,2)
(3,0,2)
(4,1,2)
(5,1,2)
(6,0,2)
(7,0,3)
(8,1,3)
(9,1,3)
(10,1,3)
(11,0,2)
(12,1,2)
(13,1,2)
(14,1,2)
(15,1,3)
(16,1,3)
(17,1,3)
(18,0,3)
(19,0,2)

real	0m0.002s
user	0m0.002s
sys	0m0.000s
psk@predator:~/Desktop/L2_18114038/src/Problem 2$
```