

Data Structure Lab

Lab Assignment - 2

Name - Radhika Garg

Roll No. - 18114060

Batch - O3

Problem1.

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 $j = (ai + b) \bmod n$. 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.

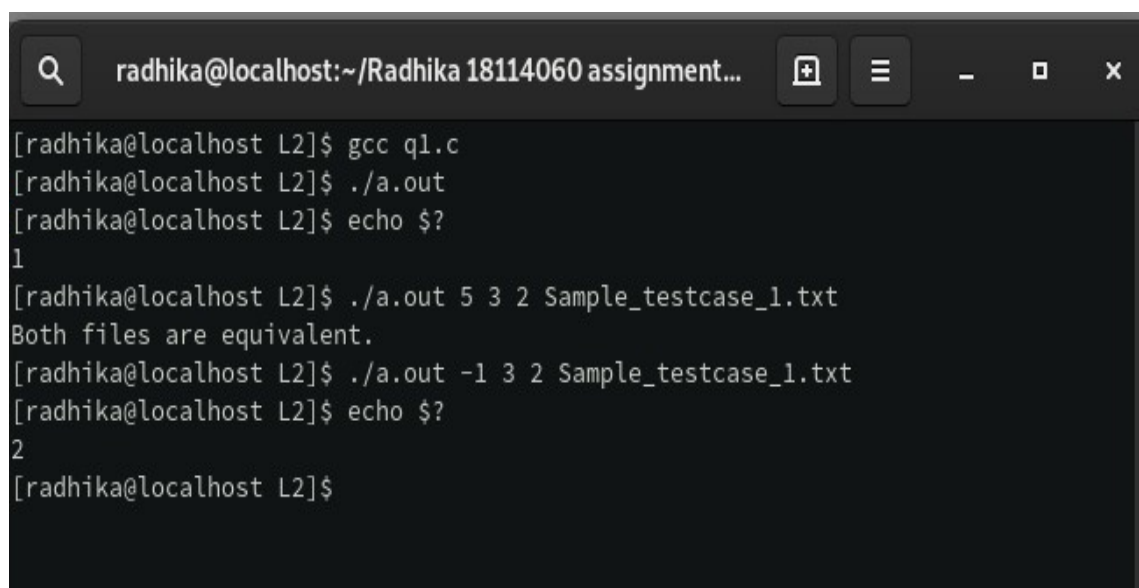
Data Structure Used:

1. A dynamic string to get what is Written in the file.

Algorithm Used:

1. Get n , a , b , c and file name in arguments.
2. If $n < 1$ the program exits with exit code 2 and for arguments less than 4 exit code is 1.
3. Add "NULL" to the string if length is not the multiple of n to make it so.
4. According to the question $j = (a_i + b) \bmod n$, where i is index of encrypted array and j for the given string.
5. Similarly decryption has i for encrypted and j for decrypted.

Screenshots:



```
radhika@localhost:~/Radhika 18114060 assignment...  
[radhika@localhost L2]$ gcc q1.c  
[radhika@localhost L2]$ ./a.out  
[radhika@localhost L2]$ echo $?  
1  
[radhika@localhost L2]$ ./a.out 5 3 2 Sample_testcase_1.txt  
Both files are equivalent.  
[radhika@localhost L2]$ ./a.out -1 3 2 Sample_testcase_1.txt  
[radhika@localhost L2]$ echo $?  
2  
[radhika@localhost L2]$
```

Here "echo \$?" displays the exit code of the previous command.
Screenshots of sample_test_case file, outputfile, decryptedfile are:



```
Sample test case  
csn-261|
```



Problem2: Medial axis transformation (MAT)

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.

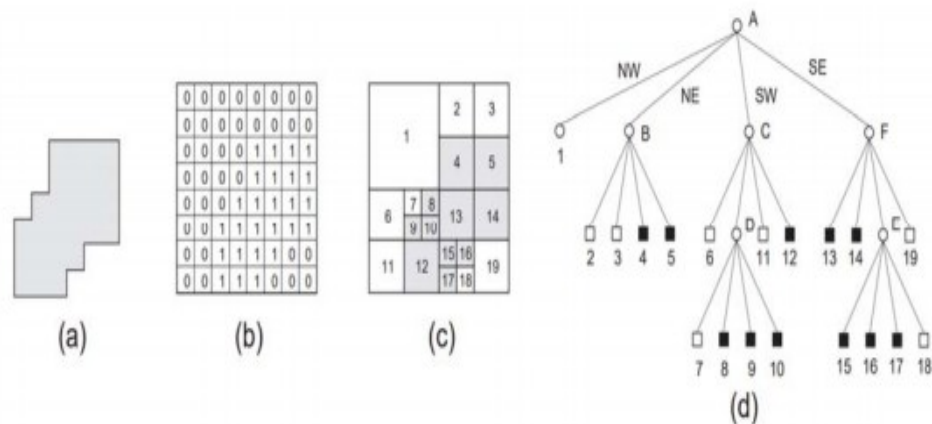


Fig 2: Flow of quadtree representation from the sample region

Example:

The above figures represent a region and its corresponding quadtree representation,

- Defined as the Sample region having all the bit value to 1.
- Defined as the binary array of size 8*8 which having bit value other than sample region within it is 0.
- 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**.
- 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 Structure Used:

1. Struct quadNode used to create the quadTree.
2. The node has 4 child nodes nw, ne, sw, se and a color 0 for white, 1 for black

Algorithm:

1. Read the input file and store the given matrix.
2. Get the next power of 2 using recursion.
3. Transform the given matrix to the bottom-right of the matrix with order 'n' that is 2^x .
4. Check for the same values if not we divide the matrix into 4 parts and then each part undergoes the same method (recursion).
5. To get the level (considering root is at level 0) use $\log(\text{size of block that n represents})$.

```
radhika@localhost:~/Radhika 18114060 assignment...  
[radhika@localhost L2]$ gcc q2.c  
[radhika@localhost L2]$ ./a.out  
The matrix:  
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  
(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)  
[radhika@localhost L2]$
```