INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

CSN-261 : Data Structures Laboratory

Assignment-2

Name : **HEMANT**

Enr. No. : **18114030**

E-mail : **hemant@cs.iitr.ac.in**

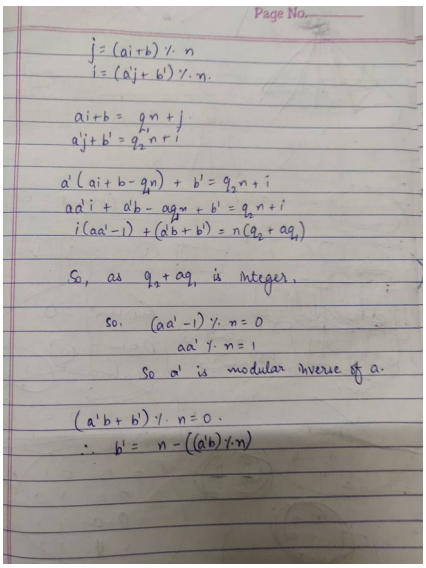
Problem 1 :

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□).

Algorithms Used :



Time taken :

1. transpose.c : 0.000562 sec
2. inverseTranspose.c : 0.000706 sec
3. compareFiles.c : 0.000262 sec

Data Structure Used :

1. Array

Snapshot :

Printing transpose.c :

```
hemant@hemant-hp:~/Desktop/Asg2/Que1$ g++ transpose.c
hemant@hemant-hp:~/Desktop/Asg2/Que1$ ./a.out 5 3 2 inputfile.txt
The encrypted string has been written in the outputfile.txt file.
Your encrypted string is :
lHleow,o r!l
d
itsh  iHsenmta
vHea uy ocpolmedt eyro uwko$r$
```

Printing inverseTranspose.c :

```
hemant@hemant-hp:~/Desktop/Asg2/Que1$ g++ inverseTranspose.c
hemant@hemant-hp:~/Desktop/Asg2/Que1$ ./a.out 5 3 2
Your decrypted string is :
Hello, world!

this is Hemant
Have you completed your work
The decrypted string has been written in the decryptedOutputfile.txt file.
```

Printing compareFiles.c

```
hemant@hemant-hp:~/Desktop/Asg2/Que1$ g++ compareFiles.c
hemant@hemant-hp:~/Desktop/Asg2/Que1$ ./a.out

------------------Comparing two files--------------------
Your equivalance is 0. GOOD WORK !!!
```

inputfile.txt

```
Hello, world!

this is Hemant
Have you completed your work
```

outputfile.txt

```
lHleow,o r!l
d
itsh  iHsenmta
vHea uy ocpolmedt eyro uwko$r$
```

decryptedOutputfile.txt

```
Hello, world!

this is Hemant
Have you completed your work
```

Problem 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. 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.

Algorithms Used :

• In this code, I have taken the input array from the txt file and then I have checked whether all elements of array are same or not.

• If same, then make all indices of same value else break array into four parts and check again and do this process till all the indices are checked completely i.e. when the array becomes maximal square array.

• Then I have made a quadtree with each node having three values as its level, label and indexValue.

• Then I have printed the maximal square array and the quadtree with all its three values.

Time taken :

0.000670 sec

Data Structure Used :

1. Array

2. Linked List

Snapshot :

Printing Maximal Square Array :

```
Enter the index of operation you want to perform :
To print the Maximal square array : 1
To print the tree nodes : 2
Exit the program : 0
Enter the index : 1

_____________________  Maximal Square Array  _____________________

01 01 02 02 05 05 06 06
01 01 02 02 05 05 06 06
03 03 04 04 07 08 11 11
03 03 04 04 09 10 11 11
12 13 16 16 22 22 23 23
14 15 16 16 22 22 23 23
17 17 18 19 24 24 25 25
17 17 20 21 24 24 25 25
_____________________________________________________________________
```

Printing Nodes :

```
Enter the index of operation you want to perform :
To print the Maximal square array : 1
To print the tree nodes : 2
Exit the program : 0
Enter the index : 2

_____________________  Tree Nodes  _____________________

(01, 0, 2)
(02, 0, 2)
(03, 0, 2)
(04, 1, 2)
(05, 0, 2)
(06, 0, 2)
(07, 1, 3)
(08, 1, 3)
(09, 1, 3)
(10, 0, 3)
(11, 0, 2)
(12, 0, 3)
(13, 1, 3)
(14, 1, 3)
(15, 1, 3)
(16, 1, 2)
(17, 1, 2)
(18, 1, 3)
(19, 1, 3)
(20, 1, 3)
(21, 0, 3)
(22, 1, 2)
(23, 0, 2)
(24, 0, 2)
(25, 0, 2)
_____________________________________________________________
```