Shared Locker

Deadline: 20/02/2022(11:59PM)

Lab 03

Course Title: Algorithm Design   Course No: **CS 222**

In this assignment you have to write a C/C++ program that implements a shared locker called *lock_wrapper.c*

Your program should have two modules, a CONFIGURE module and a USE module.

1. The CONFIGURE module should take the number of users k, a *locker_id*, and a key L as input and should generate k secondary keys $(u_1, u_2, \ldots, u_k)$ and k distinct prime numbers $p_1, p_2, \ldots, p_k$. Each secondary key should be a 4 digit number meant for each of those k users, and the primary key L is meant for the locker. You may assume $L < p_1 \times p_2 \times \ldots \times p_k$
Further, the configuration module writes an entry to the file *config.txt* where the entry is of the form:

$$locker\_id \ - \ k, p_1, p_2, \ldots, p_k$$

where each p_i is a prime number. Further, given the sequence $p_1, p_2, \ldots, p_k$, one can construct a unique L $(0 \leq L < \prod_{i=1}^{k} p_i)$ from the sequence $(u_1, u_2, \ldots, u_k)$ provided $u_i < p_i$ for $1 \leq i \leq k$.
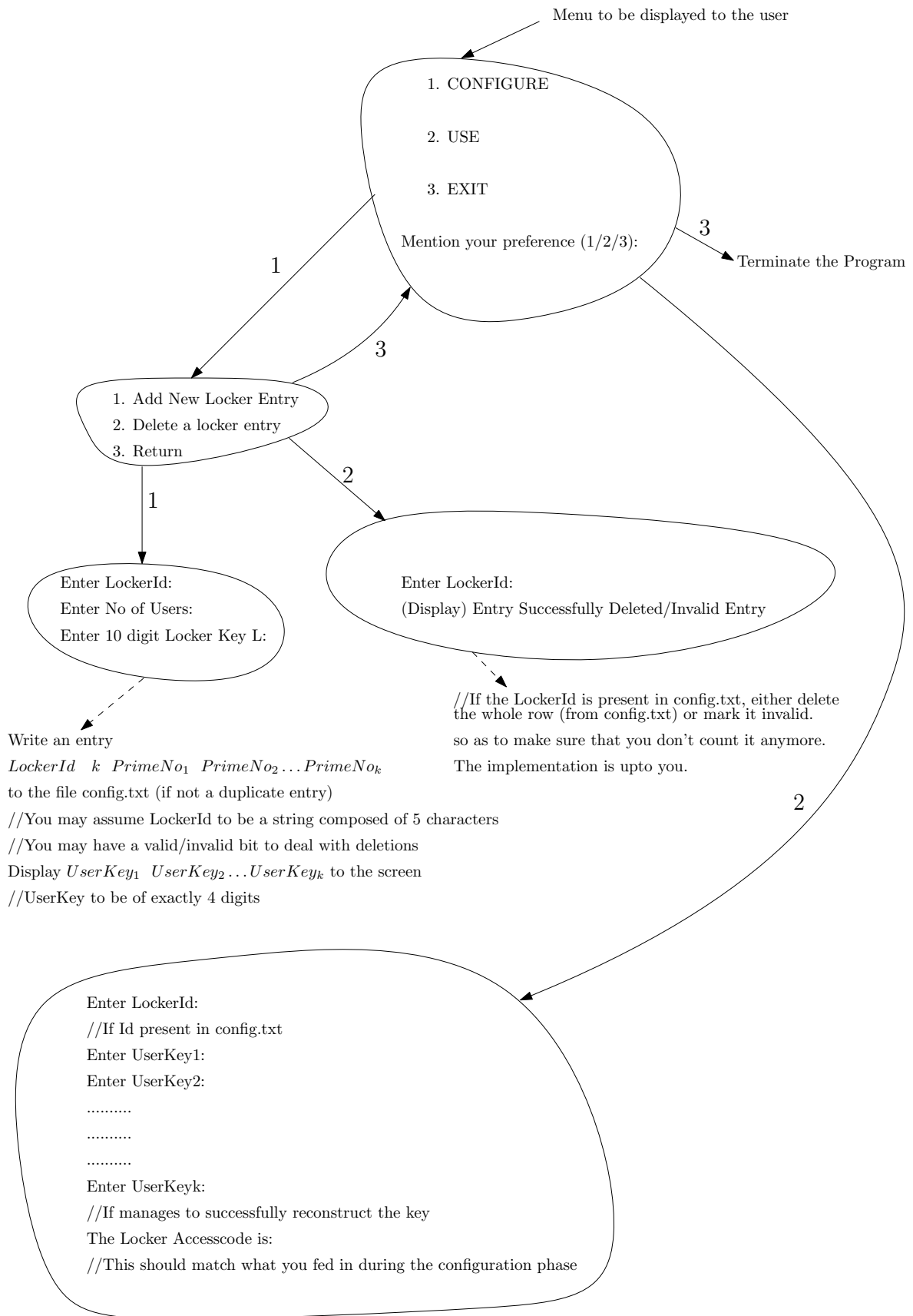
2. The USE module is supposed to read the *locker_id* and fetch the value of k and the p_is from the configuration file *config.txt* to begin with. Further, it queries the user(s) to feed in $u_1, u_2, \ldots, u_k$ (order matters) as input. The task in hand is to generate L using $p_1, p_2, \ldots, p_k$ so as to let the users access the shared locker.
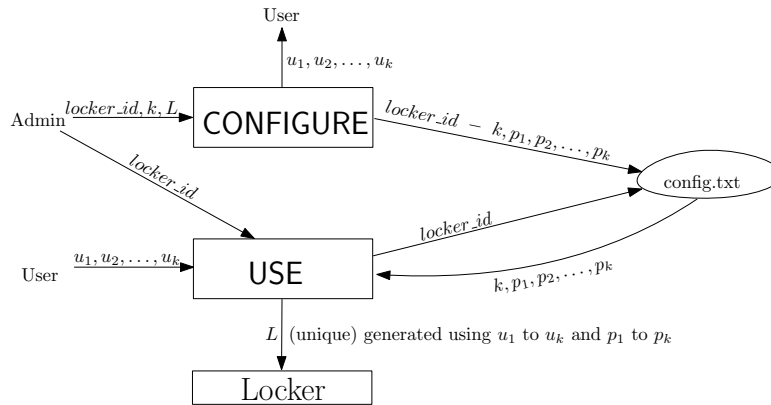
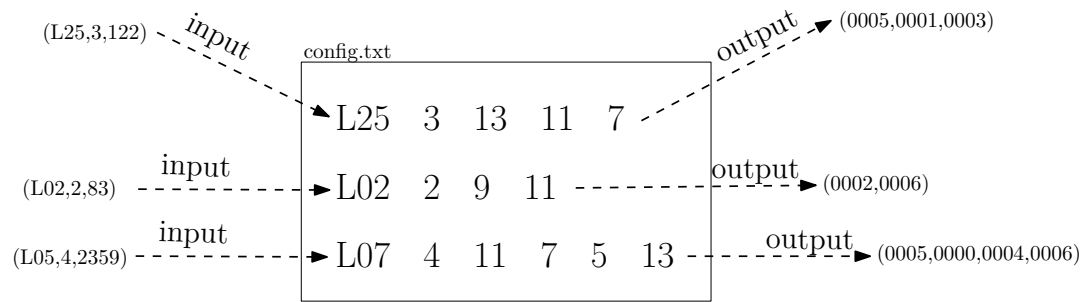Your implementation should be based on Chinese Remainder Theorem. You may assume the number of users to be 2 to begin with and pick a pair of sufficiently large prime numbers such that the product of these two prime numbers is at least 1000. Then you may try to increase it to 3, and then 4.

If your code works for at most 5 users, that's fine.

**Bonus credit for code that works for arbitrary k.

Menu to be displayed to the user

1. CONFIGURE

2. USE

3. EXIT

Mention your preference (1/2/3):

3

Terminate the Program

1

3

1. Add New Locker Entry
2. Delete a locker entry
3. Return

2

1

Enter LockerId:

Enter No of Users:

Enter 10 digit Locker Key L:

Enter LockerId:

(Display) Entry Successfully Deleted/Invalid Entry

//If the LockerId is present in config.txt, either delete the whole row (from config.txt) or mark it invalid.

so as to make sure that you don't count it anymore.

The implementation is upto you.

Write an entry

$LockerId \quad k \quad PrimeNo_1 \quad PrimeNo_2 \ldots PrimeNo_k$

to the file config.txt (if not a duplicate entry)

//You may assume LockerId to be a string composed of 5 characters

//You may have a valid/invalid bit to deal with deletions

Display $UserKey_1 \quad UserKey_2 \ldots UserKey_k$ to the screen

//UserKey to be of exactly 4 digits

2

Enter LockerId:

//If Id present in config.txt

Enter UserKey1:

Enter UserKey2:

....

....

....

Enter UserKeyk:

//If manages to successfully reconstruct the key

The Locker Accesscode is:

//This should match what you fed in during the configuration phase

2

Sample *config.txt* file

Diagram showing config.txt file with inputs and outputs:

Inputs:
- (L25,3,122) — input
- (L02,2,83) — input
- (L05,4,2359) — input

config.txt contents:
```
L25   3   13   11   7
L02   2   9   11
L07   4   11   7   5   13
```

Outputs:
- (0005,0001,0003) — output
- (0002,0006) — output
- (0005,0000,0004,0006) — output

Additional Instructions

1. You may assume the locker key L to be atmost 10 digits long.
2. Each user key should exactly be composed of 4 digits.
3. The number of users k to be atmost 8.
4. You may use space or tab as the delimiter.
5. You don't have to stick to the front end template as such. May improvise on your own.

*************END*************