MULTITHREADING WITH PTHREADS

Q1 [Primes in a Parallel way!]
Given randomly generated N positive numbers as input, find out the highest valued prime number. Your program should take two arguments, i.e., *number of elements* and *number of threads* as command line parameters. In your program (main function) should generate random inputs before creating threads as shown in class example (refer find_max_p.c). Test the correctness and performance using different values for number of threads and number of elements. *Submit only the C source file in moodle*.

Q2 [A simplified bank!]
A hypothetical bank has asked you to write a software to carry out their day end transaction in an efficient manner. The bank provides you with the status of the accounts before the day's business start and a log of transactions throughout the day. Your program should update the status of accounts for the next day.

Specifically, there are two input files (one containing account status (say acc.txt), another containing all transactions (say txn.txt)) passed as command line to your program. Additionally, two more arguments specifying the number of threads and number of entries in transaction file are also passed to your program. We should be able to compile and execute your program as follows,

bash$ gcc q2.c -lpthread -o q2

bash$ ./q2 {account file e.g., acc.txt} {Transaction file e.g., txn.txt} {#of transactions} {#of threads}

Your program should output the updated account status in console in the format same as the acc.txt file.

Format of the two files are as follows (no file headers in the actual input),

acc.txt

A/C No	Balance
1001	25000.40
1002	9000.90
---	---
---	---
11000	9834.00

For convenience, assume that there are 10000 bank accounts with A/C numbers starting from 1001. Note that the heading (underlined A/C No and Balance) is not part of the file

txn.txt

Transaction Seq	Type	Amount	A/C 1	A/C 2
1	1	1000.30	7456	0
2	2	2000	6454	0
3	3	0	9484	0
4	4	5000	7456	3212
5	1	4000	8746	0
- --	---	----	---	----
----	---	----	---	---

Transaction logic are as follows,

Type = 1) It is a deposit transaction. Amount mentioned is added to the balance of account (A/C 1) after subtracting a 1% service charge on the deposit amount.

Type = 2) It is a withdrawal transaction. Amount mentioned is subtracted from the balance of the account (under A/C 1) along with an additional 1% service charge on the withdrawal amount.

Type = 3) An interest of 7.1% is added to the mentioned account under (A/C 1)

Type = 4) This is a fund transfer transaction. Mentioned amount is transferred from A/C 1 to A/C 2. A service charge 1% of the transferred amount is levied on the source account as well as the destination account.

Your program should read the account status from the accounts file (e.g., acc.txt) and initialize the accounts (may be as an array of structures). You should also read the transactions file (e.g., txn.txt) into memory completely before invoking the threads as shown in class example (refer block_hash.c).

Notes:

1. Do not assume that the transactions are uniformly distributed across all the accounts. So, **do not split** the transactions across threads on the basis of A/C numbers.
2. Incorporate as much parallelism as possible. For example, when a thread is updating one particular A/C, blocking parallel operations on all A/Cs should be avoided.
3. Test your implementation thoroughly before submission.

Submission format

Your submission should be a single archive (tar, zip etc.) named after your roll no. (<roll_no.zip>) which expands to a folder containing two bash source files (qn1.c and qn2.c).