CSCI 3232 Systems Software Assignment 7

Upload all your files to the correct dropbox in Folio before the deadline --- **11:30PM, Apr 12, Sunday, 2020.**

Note: You need to be able to figure out whether there are race conditions in a program and tell the output without actually running the code. You should also be able to apply mutex and semaphore techniques to synchronize your programs should any race conditions arise.

1. (18 pts) Read given code RaceOrNot1.c and write all possible outputs of the program. Assume there will be no thread creation or joining failures or mutex failures. If you believe there is only one possible output, you just need to write that output.
2. (18 pts) Read given code RaceOrNot2.c and write all possible outputs of the program. Assume there will be no thread creation or joining failures or semaphore failures. If you believe there is only one possible output, you just need to write that output.
3. (18 pts) Read given code RaceOrNot3.c and write all possible outputs of the program. Assume there will be no thread creation or joining failures or semaphore failures. If you believe there is only one possible output, you just need to write that output.
4. (30 pts) From previous homework you are already familiar with the math function *f* defined on positive integers as f(x)=(3x+1)/2 if x is odd and f(x)=x/2 if x is even. Given any integer *var*, iteratively applying this function *f* allows you to produce a list of integers starting from *var* and ending with 1. For example, when *var* is 6, this list of integers is 6,3,5,8,4,2,1, which has a length of 7 because this list contains 7 integers (call this list the Collatz list for 6). Write a C or C++ program **A7p4.c**(**pp**) that accepts one command line argument which is an integer *n* between 2 and 6 inclusive. Use pthread to create *n* threads to count how many integers between 1 and 60 inclusive have their Collatz list length (a) less than or equal to 8; (b) greater than 8 and less than 16; (c) greater than or equal to 16. You should divide this list generation and length calculating task among the *n* threads as evenly as possible. For example, if *n* is 3, then each thread is supposed to process 60/3=20 Collatz lists. Print out the three numbers representing the counts for (a)(b)(c) mentioned above. Use mutex or semaphore to avoid race conditions if necessary. Note: if you do not use pthread to divide the task among the threads, you may get zero points. A sample run of the compiled program A7p4 is shown below. You do NOT need to submit screen shots. Instead submit your source file together with a working makefile to compile this program as separate files.

[kwang@computer][~/temp]$./A7p4 3

using 3 threads.

the number of integers from 1~60 whose Collatz list has length <=8 is 16

the number of integers from 1~60 whose Collatz list has length between 8 and 16 exclusive is 26

the number of integers from 1~60 whose Collatz list has length >=16 is 18

1. (16 pts) (1) Summarize in your own sentences what are the classical synchronization problems (producer & consumer problem, readers & writers problem, dining philosophers problem). (2) Understand why the four conditions for a deadlock are sufficient and necessary for it to occur. Submit your descriptions of the three synchronization problems and the four conditions for a deadlock.

Checklist of files to be submitted: A7p4.c(pp), makefile, solution file for problems 1,2,3,5.