CS6233

Midterm Exam – Part2

Programming Problem

1 hour, 25 minutes

10 points

Please submit your C-code for the following programming problem via the assignment page in Brightspace before the allotted time expires.

This problem is **open book, notes, class page, man pages, etc.**

You must **work alone** and may not consult or seek help from others; locally, remotely, via internet forums, etc.

You may perform your work on a Linux virtual machine or a Linux laptop.

You shall checkout with the one of the proctors before exiting the exam.

Programming problem:

Write a program that uses a Monte-Carlo simulation to estimate the probability of two students in a class having the same birthday. The program shall split the simulation over four processes.

Your program's main process shall create 3 child processes (NOT threads) to speed up the computations, such that all the child processes have a **common parent, which is the main process**, as shown in the figure below. All processes (the parent and the three children, a total of four) shall share only one variable (an integer, lets' call it `nhits`) and the corresponding synchronization variable (e.g., a semaphore). **Do not use atomic integers**.

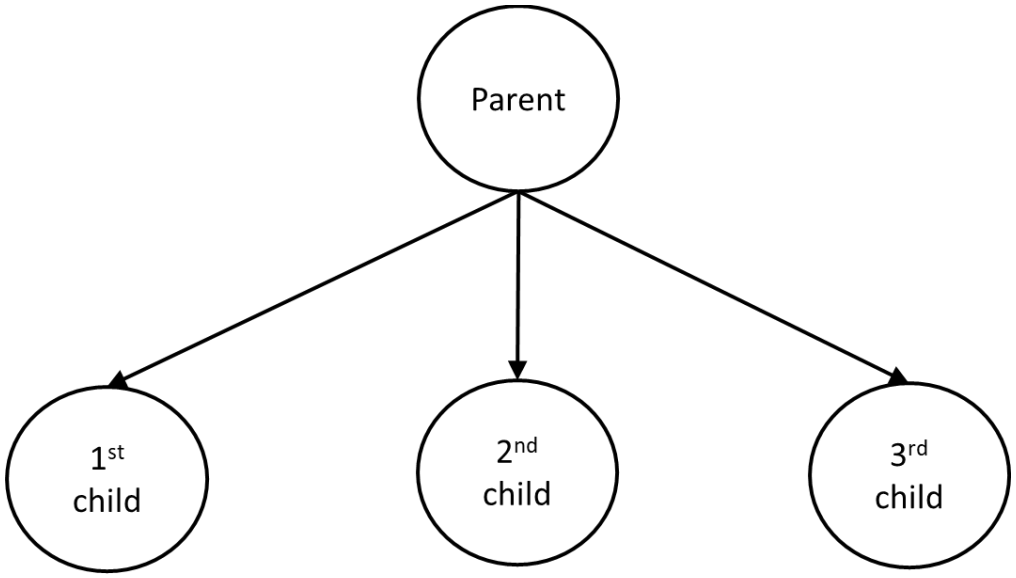

Fig. (1) - Process tree

Each process shall perform a number of trials/experiments NUM_TRIALS=100,000 (defined as a macro), in which it creates a list of n random numbers, each has a value between 0 and 364 representing each person's birthday within the year. If (at least) two of them coincide, the thread increments a **shared** variable `nhits` by 1 (for each trial). The shared variable holds the total number of times the experiments/trials succeeded (i.e., 2 or more students had a matching birthday). *n* is passed to your program as an input argument, you should test with n=23 and n=49.

Note that each process must immediately increment the shared integer variable (nhits) **immediately after evaluating a trial and NOT wait for the entire NUM_TRIALS to be computed**.

When all child processes complete, they then exit. When the parent process completes its computations, it then waits for all the children to exit and then it calculates and output the probability as:

```
nhits/(4*NUM_TRIALS)
```

For a class of 49 students, we expect the number to be about 97%, for 23 students, it should be about 50% (https://en.wikipedia.org/wiki/Birthday_problem).

After the parent process prints the probability, it must destroy the semaphore(s) and the shared memory.

Some useful notes:

- Each process should seed the random number separately by calling srand(time(NULL)). There's no concern about thread safety since each call is from a different single-threaded process.