CSCI 340 Homework 1
Spring 2018

Collaboration: None
Due Date: 12:30 or 1:30 pm Feb 9
Points: 50

1 Objectives

For this assignment you will be developing a C program to run on a Linux system. This assignment will allow you to gain experience in the following areas:

- **Unix Development**: Writing, compiling, executing, and debugging a C program
- **Basic Programming Concepts**: Declaring variables, arrays, using/defining data types, using pointers, operators, expressions, selection statements, looping statements, functions, and header files
- **Input/Output (IO)**: Getting input from the command line, displaying information to the console
- **Recursion**: Writing a recursive function
- **Processes**: Create a process, waiting on a process to terminate and releasing resources when a process terminates

2 Provided Files

The two files listed below are provided to you:

- **fib_fork.c**: This complete C program is an example of computing Fibonacci using recursion and *fork()*.
- **BinaryParity.java**: This complete Java program is an example of computing the *binary parity* of a number using recursion. An integer has even parity if its binary representation contains an even number of one bits; otherwise it has odd parity.

Both files are well commented.

3 Todo

Look at, run, and understand what the simple *binary parity* recursive Java program is doing. In particular, note how it is getting input from the command line, computing the parity using recursion, and how it is outputting the result using a string array.
Look at, run, and understand what the simple *fib_fork* C program is doing. We will discuss this in class, but you will need to put some time into running it with different input values and outputting resulting intermediate values during the recursion. It is also important to see how each process is returning a value via *exit()*. Do a "man" on each of: *fork(), waitpid(), WIFEXITED(), WEXITSTATUS(),* and *exit()*.
From your understanding of both *binary parity* and *fib_fork()*, write a recursive solution in C for the *binary parity* problem that uses processes. Your C solution must use recursion, processes, and in general behave similar to *fib_fork()*. However, of course, the problem you will be solving is not the Fibonacci problem, but rather the recursive *binary parity* problem.
Specifically, you must get input from the command line in C, check for proper number of input parameters, parse the integer input, check that the input integer is non-negative, use a constant static array for printing the "even" or "odd" parity result — all, similar to what was done in *BinaryParity.java*. Again, the recursion must be done using processes — similar to *fib_fork.c*.
Call your program, *parity_fork.c*.

4 Collaboration and Plagiarism

This is an **individual assignment**, i.e. **no collaboration is permitted**. Plagiarism will not be tolerated. Submitted solutions that are very similar (determined by the instructor) will be given a grade of zero. Please do your own work, and everything will be OK.

5 Submission

Create a compressed tarball, i.e. *tar.gz*, that only contains the completed *parity_fork.c* file. The name of the compressed tarball must be your last name in lower case. For example, *ritchie.tar.gz* would be correct if the original co-developer of UNIX (Dennis Ritchie) submitted the assignment. Only assignments submitted in the correct format will be accepted (no exceptions). Submit the compressed tarball to the appropriate Dropbox on OAKS by the due date. You may resubmit the compressed tarball as many times as you like, only the newest submission will be graded.
To be fair to everyone, late assignments will not be accepted. Exceptions will only be made for extenuating circumstances, i.e. death in the family, health related problems, etc. You will be given a week to complete this assignment. Poor time management is not excuse. Please do not email assignment after the due date, it will not be accepted. Please feel free to setup an appointment to discuss the assigned coding problem. I will be more than happy to listen to your approach and make suggestions. However, I cannot tell you how to code the solution. Additionally, code debugging is your job. You may use the debugger (gdb) or print statements to help understand why your solution is not working correctly, your choice.

6 Grading Rubric

For assignments that compile and run without faulting, the grading rubric is provided in the table shown below.

checking command line input	5 points
parsing command line input	5 points
using a constant static array for output	5 points
properly using fork with recursion	35 points

If the assignment does not compile or faults early with little or no output, then anywhere from 10 to 50 points will be taken off based on code inspection and/or code modification on my part. During grading I typically try to fix some errors. But, I will *not* try to "fix" a cascading list of errors in your code. Having so many errors demonstrates incomplete effort on your part towards developing and testing a correct solution.
That being said, I'm available to meet with you and discuss your lab. If you feel there is more to your solution, you can "prove" this to me (and redeem some of the points taken off) by demonstrating a correct solution taken from your submitted solution, and showing me all the differences in the code you made. You must do so within a week of my feedback on the assignment, and, prior to any solutions being handed out for this assignment.