CPS 310 final exam, 12/12/2014

Your name please: ____________________________

To celebrate completing my five straight semesters of CPS 310, and in keeping with the holiday spirit, the theme of today’s exam is **A Few of My Favorite Things**. These are some of my favorite problems from recent exams. So if you studied the recent exams, then you hit the jackpot. (Of course I may have changed them slightly here and there.)

**Part 1. Fun with forks**

What is the output generated by this program? (Assume that there are no errors or failures.) [30 pts]

```c
int main()
{
    int i = 1;
    if (fork() != 0) {
        i = i + 1;
        if (fork() != 0)
            exit(0);
    }
    fork();
    printf("%d\n", i);
}
```

*Optional:* briefly justify/explain your answer (for partial credit if your answer is wrong).
Part 2. Fixing the plumbing

For some command lines the shell (e.g., dsh) spawns two child processes connected by a pipe.

The following pseudocode proposes sequences of system calls for the parent shell and for each child to set up the pipe. Naturally, the sequences are incorrect. Point out three key errors in the sequences. For each error, write a sentence or two explaining why it is wrong and how to fix it. [30 points]

Extra credit: estimate how many hours of sleep each error will cost you. Justify your answer without reference to “tcsetpgrp” or “setpgid”.

```
parent
fork();
wait*(…);
fork();
wait*(…);

left child
pipe();
close(…);
dup2(…);
exec*(…);

right child
close(…);
dup2(…);
exec*(…);
```
Part 3. You tell me

A few weeks ago someone on Piazza asked me to explain the difference between monitors (mutexes and CVs/condition variables) and semaphores. One way to understand the difference is to implement each abstraction in terms of the other. We did that for one direction in class. Let’s try the other direction.

The pseudocode on the right is a “first cut” implementation of monitors using semaphores. What’s wrong with it? Write a list of three distinct correctness properties that this solution violates even when used correctly. Outline a fix for each one. Feel free to mark the code. [30 points]
Part 4. Sleeping Professor

A certain Duke professor likes to sleep in the Perfect Chair™ in his office when there are no students around. The professor wakes up when students visit the office to ask questions.

Write two procedures to synchronize threads for the professor and students. When the professor thread has nothing to do, it calls `IdleProf()`. `IdleProf` sleeps if there are no students waiting, otherwise it signals one student to enter the office, and returns. A student with a question to ask calls `ArrivingStudent()`, which waits until the professor is idle; if no students are waiting, then the student wakes up the sleeping professor.

The idea is that the professor and exactly one student will return from their procedures “at the same time”. (They then discuss a topic of mutual interest, then the student goes back to studying and the professor calls `IdleProf` again.)

Implement `IdleProf` and `ArrivingStudent` using monitors or semaphores: your choice. [40 points]

```c
IdleProf() {

}
```

```c
ArrivingStudent() {

}
```
Part 5. Android

These questions pertain to concepts of Android as discussed in class. Answer each question in the space available, focusing on the high points. Use the “lingo”. Please keep it short! [50 points]

(a) Why is it important to not write code that blocks the UI/main thread? When the UI thread does block, what does it wait for?

(b) How does Android determine the user ID for an app process? How is this different from Unix/Linux?

(c) What does Android do if an app fires an intent at an app or component that is not active? (You may assume that the sender has permission to fire the intent.)

(d) A binder call executes stub methods on each side of the process boundary. How does the system “know” which stub methods to execute? (Or: where do they come from?)

(e) Android maintains a reference count for each component. What does Android do if a reference count transitions to zero?
Part 6. File/storage systems

These questions pertain to the nexus of file systems and virtual memory. Answer each question in the space available, focusing on the high points. Use the “lingo”. Please keep it short! [50 points]

(a) Why are file block maps skewed? By “skewed” I just mean that the maps form a tree that is unbalanced, in contrast to a hierarchical page table, which is balanced.

(b) Why might an interrupt handler wake up a thread? (Give three examples.)

(c) Sometimes a page fault handler installs a zero-filled page rather than reading a missing page from disk. Under what conditions does the system do this?

(d) Why does mirroring (RAID-1) improve throughput for random block reads but not for writes?

(e) Why does striping-with-parity (RAID-5) have lower throughput for random block writes than pure striping (RAID-0) does?
Part 7. Cryptosystems [60 points]

(a) Alice and Bob are back together. Today Alice wants to send Bob a message that is secret and also authenticated, so that Bob "knows" the message came from Alice. Alice and Bob have keypairs and each knows the other's public key.

How should Alice send the message? How should Bob validate the message? Briefly explain why your scheme works, and note any additional assumptions.

(b) One-way hash functions are fast and cheap. But what are they good for? I am looking for two specific examples of how they are used in practice, explaining how their properties are useful for those examples.

(c) Often a server returns a certificate in response to a connect request. The certificate contains the server's public key. Presuming that the certificate is valid, what does the client do with the server's public key?
Part 8. Extra credit

(a) Did you fill out an online course evaluation?

(b) What is an abstraction? Feel free to illustrate with a picture and/or examples from the class.