We'll post an outline of each recitation after the fact so you know what was covered in case you had to miss class that day. The recitation outlines are not a full replacement for recitation!

This week's recitation was largely a problem-solving session (good for exam prep). These are the problems we went over; some sections may not have gotten to every problem.

Naming

In general, we can say that some name N *resolves* to some object X. Reversing that, we can say that an object X *is named by* some name N. In this question, we consider name/object relationships in DNS and Unix. Although there are many kinds of names in systems, we are focusing here on human-friendly names represented as legible character strings.

Pick True or False for each of the following statements:
a. True / False DNS allows a single IP address to be named by multiple names.
b. True / False DNS allows a single name to resolve to multiple IP addresses.
c. True / False Unix allows a single file to be named by multiple names.
d. True / False Unix allows a single name to resolve to multiple files.
e. True / False A DNS client can ask a DNS server to resolve a given host name to an IP address.
f. True / False A DNS client can ask a DNS server to change which IP address is named by a given host name
g. True / False A Unix program can ask the file system to resolve a given file name to a file
h. True / False A Unix program can ask the file system to change which file is named by a given file name.

UNIX - Filesystems

Consider this scenario:

- A process creates and writes a new file named "/u/katrina/docs/foo" in the UNIX file system (consider that a call to create())
- After creation, the process writes 17 bytes into this file (consider that a call to write())

How does this work? Draw a picture!

UNIX - Processes

a. True / False If you follow a shell command with &, the shell will make a new process in which to run the command while the shell goes on in parallel to read and execute the next command.
b. True / False If you don't follow a shell command with &, the shell will not create a new process but rather run the command to completion in the shell process.

c. True / False Checking the return value of the fork () function enables a child process to execute different instructions from its parent.
d. True / False Since a child process can write to all files that are open for its parent at the time of the fork, race conditions are possible where both parent and child are writing to the same file at roughly the same time.

Virtual Memory

Suppose program X has the following page table entries that map virtual to physical pages. Here, W is the read/write bit and P is the present bit.

Virtual address	**Physical address**	**W**	**P**
1	4	1	1
2	1	1	0
3	2	0	1

All memory is initialized with 0 values. X executes the following three write calls (the syntax is write(a, b), where a is an address and b is a value)

```
write(1, 10)
write(2, 20)
write(3, 30)
```

Which of the calls will cause an exception, if any?

After X terminates, what are the contents of physical addresses 1, 2, 3, and 4?

Bounded Buffers

Ben has the job of building locks to avoid race conditions in his team's new operating system, TypOS. He figures that he will do this with a simple character variable, with "x" meaning the lock is available and "y" meaning the lock is held.

Ben's version of release() is:

```
release(lock):
        lock = "x"
```

And Ben's version of acquire() is:

```
acquire(lock):
        while lock != "x"
                do nothing
        lock = "y"
```

What is wrong with Ben's implementation? Exactly one of the below options is correct. For the ones that are wrong, explain why they are *not* correct.

a. Ben should be using integers, not characters
b. He should be testing the value of lock in release()
c. He should be testing the value of lock in release() and *not* testing the value of lock in acquire()
d. He needs an atomic swap or exchange to implement acquire()
e. He should initialize locks with a third value that is neither "x" nor "y"