Name:

Operating Systems
Fall 2015
Test 3
December 2, 2015

Closed books, notes, cell phones, PDAs, iPods, laptops, etc. No headphones, please. You do not need to use a calculator.

You have 75 minutes to solve 5 problems. You receive 11 points bonus for making it so far. As with any exam, you should read through the questions first and start with those that you are most comfortable with.

Partial credit will be offered only for meaningful progress towards solving the problems.

Please read and sign below if you agree with the following statement:

In recognition of and in the spirit of the academic code of honor, I certify that I will neither give nor receive unpermitted aid on this exam.

Signature:_____

0	11/11
1	/15
2	/20
3	/16
4	/20
5	/18
Total	**/100**

1. **Disk:**

Sketch rough graphs ("back of napkin") illustrating the relationships among the following quantities. Note any important assumptions underlying your analysis.

a. Disk access latency as a function of rotation speed.

b. Number of accesses to the disk as a function of the degree of multiprogramming.

c. Peak disk read bandwidth and read latency as a function of the size of the read request (number of blocks requested).

2. Disk scheduler fun [20 points, 4 each]

This question examines a disk's internal scheduler as a black box, to see if we can learn anything about its behavior. The inputs we give to the disk are a bunch of requests; the outputs we observe are the order in which the requests are serviced. For example, we might issue requests to blocks 0, 100, and 200 at the same time. The disk might decide to service 100, then 200, and then 0, depending on its internal scheduling algorithm.

We also know some details of the disk. It has one surface with 100 tracks, each of which has 100 sectors. The outer track (track=0) contains sectors 0...99, the next track (track=1) contains sectors 100...199, and so forth.

In this problem, each question specifies a workload sent to a particular disk; you are then asked to determine the possible disk scheduling policy based on the order in which the requests are completed. In other words, what can you say about the algorithm in use? What about the initial state of the disk, before the requests arrived?

(a) Disk Model A
Requests: 0, 1, 2, 3, 4, 5
Completed in order: 0, 1, 2, 3, 4, 5
What do you think Disk A's scheduler is doing? What might the initial state of the disk have been?

(b) Disk Model B
Requests: 0, 500, 200, 400, 300, 100
Completed in order: 0, 100, 200, 300, 400, 500
What do you think Disk B's scheduler is doing? What might the initial state of the disk have been?

(c) Disk Model C
Requests: 0, 500, 200, 400, 300, 100
Completed in order: 200, 300, 400, 500, 100, 0
What do you think Disk C's scheduler is doing? What might the initial state of the disk have been?

(d) Disk Model D
Requests: 0, 50, 110, 600
Completed in order: 0, 110, 50, 600
What do you think Disk D's scheduler is doing? What might the initial state of the disk have been?

(e) Could these disks (A through D) actually be using the same scheduler? If so, why? If not, why not?

3. **RAIDs. [16 points, 2 each]**

For this question, we'll examine how long it takes to perform a small workload consisting of 12 writes to random locations within a RAID. Assume that these random writes are spread "evenly" across the disks of the RAID. To begin with, assume a simple disk model where each read or write takes D time units.

a. Assume we have a 4-disk RAID-0 (striping). How long does it take to complete the 12 writes?

b. How long on a 4-disk RAID-1 (mirroring)?

c. How long on a 4-disk RAID-4 (parity)?

d. How long on a 4-disk RAID-5 (rotated parity)?

e. Now assume we have a better disk model, in which it takes S time units to perform a random seek and R units of time to perform a full rotation; assume transfer is free. How long do the 12 random writes take to complete on a 4-disk RAID-0?

f. How long on a 4-disk RAID-1 (mirroring)?

g. How long on a 4-disk RAID-4 (parity)?

h. How long on a 4-disk RAID-5 (rotated parity)?

4. A Basic File System. [20 points, 5 each]

For this question about basic file systems, assume a simple disk model where each disk read of a block takes D time units. Also assume the basic layout is like the very simple file system or the Fast File System.

a. Assume that all data and metadata are initially only on disk. Assume further that all inodes are in separate blocks, and that each directory is only one block in size. How long does it take to open the file /a/b/c/d.txt?

b. Assume after opening the file, we read the file in its entirety. It is a big file, containing 1036 blocks. The inode itself has room for 12 direct pointers and 1 indirect pointer. Disk addresses are 4 bytes long, and disk blocks are 4KB in size. After opening it, how long does it take to read the entire file?

c. Now assume a new inode structure is introduced, in which there is only one pointer: a double indirect pointer, which points to the double indirect block, which can point to 1024 indirect blocks, each of which can point to 1024 blocks. After opening the file, how long does it take to perform 50 random reads within a very large file?

d. How long does it take to close a file, approximately (assuming disk accesses are the dominant cost)?

5. **Crash and File System Consistency**: **[18 points, 3 each]**

a. Fsck is the Unix file system crash recovery program. For each of the following fsck error messages, what do you believe fsck has seen in the crashed file system to generate the error message:

   i. File's inode link count is 1, should be 2.

   ii. Free bitmap entry for block 3323 is 0, should be 1 (allocated).

   iii. Cylinder group 4 - free block count is incorrect.

b. Which of the following conditions would likely represent a serious problem with a file system (Justify answer for each.)

   i. A write to a data block that contains no on-disk inodes pointing at it.

   ii. A write to a data block that contains multiple on-disk inodes pointing at it.

   iii. A write to a data block that is marked as free in the on-disk bitmap.