CS-245 Database System Principles – Winter 2004
Assignment 1

Due in class on Thursday, January 15.

- State all assumptions.
- Assignment policies are listed at *http://cs245.stanford.edu/policies.html*
- You can email questions to **cs245-staff@cs.stanford.edu**.

Problem 1 (20 points)

Consider a hard drive with 16 magnetic surfaces. Each surface has 8192 tracks. The disk rotates at 7200 RPM, and has a usable capacity of 4 gigabytes (4×2^{30} bytes). Assume 5% of each track is used as overhead.

a. What is the burst bandwidth this disk could support reading a single block from one track?
b. What is the sustained bandwidth this disk could support reading an entire track?
c. What is the average rotational latency, assuming it is not necessary to start at the beginning of the track?
d. Assuming the average seek time is 8 ms, what is the average time to fetch a 4 kilobyte (2^{12} bytes) sector?

Problem 2 (15 points)

Consider the Megatron 747 disk with the following properties:

- There are four platters providing eight surfaces.
- There are 10,000 tracks per surface.
- There are (on average) $2^8 = 256$ sectors per track.
- There are $2^9 = 512$ bytes per sector.
- The disk rotates at 5400 RPM.
- The block size is $2^{12} = 4096$ bytes
- Assume 10% of each track is used as overhead.
- The time it takes the head to move n tracks is $(1 + n/500)$ milliseconds.

Suppose that we know that the last I/O request accessed cylinder 2000.

a. What is the expected (average) number of cylinders that will be traveled due to the very next I/O request to this disk?
b. What is the expected block access time for the next I/O, again given that the head is on cylinder 2000 initially?

Problem 3 (20 points)

Suppose that we are scheduling I/O requests for a new Megatron 747 disk with 8192 cylinders. Assume that the average latency and transfer times for the disk are 6.5 and 0.5 milliseconds (respectively), and that the seek time is (1+*n*/500) milliseconds. Initially the head is at cylinder 4000, and then the following requests come in:

	Time	Request
1.	0 milliseconds	Request for block on cylinder 6500
2.	3 milliseconds	Request for block on cylinder 2000
3.	11 milliseconds	Request for block on cylinder 8000
4.	19 milliseconds	Request for block on cylinder 3500
5.	26 milliseconds	Request for block on cylinder 4500

a. If we use the elevator scheduling algorithm, state for each request what time it is serviced completely. Assume that once the disk controller issues an IO request, the request cannot be changed.
b. If we use a first-come-first-served scheduler, state for each request what time it is serviced completely.

Problem 4 (30 points)

A *mirrored* disk consists of two disks with identical content (same block layout). Mirrored disks are useful in high-availability applications: if one disk fails, the data is available from the other disk. Writes to mirrored disk are more expensive, since two IO actions must be preformed (the same data block is written to the same location on each disk). However, reads can be preformed more efficiently, since a particular block can be read from one or the other disk.

For this assignment, please write (see below) an efficient algorithm for scheduling reads (no writes) on a mirrored disk. Within each disk, use an elevator-like scheduling strategy. Use the following variables to represent the state of the disks:

- AIdle: True if disk A is idle, false otherwise.
- BIdle: True if disk B is idle, false otherwise.
- ANext: Block where current IO for disk A will end (or current block if AIdle is true). As in Problem 3, assume that once an IO is started, it cannot be interrupted nor its destination block changed.
- BNext: Block where current IO for disk B will end (or current block if BIdle is true).
- N: Number of blocks on each disk. Blocks range from 1 to N.
- Queue(i): The waiting IO requests. For each entry, Queue(i) is the target block that must be read.
- M: Number of queued requests; if M is 0, then there are no requests waiting.

You only need to write one of the scheduler procedures, mainly the procedure that is called when an IO terminates. Procedure DoneIO (which you write) is called by the disk controller when an IO finishes, either on disk A or on disk B. At this point, either AIdle or BIdle (or both) must be true. Once your procedure has decided what IO is next, it calls on procedure PerformIO(D, k),

which initiates a read of block Queue(k) in disk D (where D is A or B). Procedure PerformIO(D, k) also deletes entry k from the queue, and sets the appropriate Idle flag to false.. You do not have to write procedure PerformIO.

Your procedure DoneIO should be clearly written and should contain useful comments. You will be graded on (a) clarity, (b) correctness and (c) efficiency of the procedure, in this order. That is, you will get more points for a clearly written procedure that is not terribly efficient, as opposed to one that is very efficient but we cannot decipher.

Problem 5 (15 points)

You are designing a file system for a medical application. Each patient record has 15 fields that always occur (e.g., name, patient number) and 25 fields that may or may not be relevant or known for a patient (e.g., number of children given birth to, cholesterol level, etc). Assume that each of the optional fields is relevant or known for a particular patient with probability *p*. For the required fields, the values are stored in a fixed 40 bytes. For the optional fields, the values are stored in a fixed 10 bytes.

You are considering two options:

i. A fixed format record
ii. A variable format record where all optional fields are tagged. Each tag is two bytes. The sequence (possibly empty) of optional fields is terminated with a special two byte tag.

a. What is the expected size of a record for each option? (Your answer may be a function of *p*.)
b. For what range of *p* value is the fixed format option best?