DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033—Computer System Engineering Handout 32—May 7, 1996

Quiz 3 Sample Questions

Here is the FAA (Frequently Answered Answers) for Quiz 3:

1. Quiz 3 will be from **2–3pm** on **Wednesday, May 15, 1996**.

2. There are ***three*** rooms. Go to the room that corresponds to your last name:

4-270	<--->	A to H
4-370	<--->	I to R
2-190	<--->	S to Z

3. The quiz will cover all the material from Lecture 17 to Recitation 24: storage, fault tolerance (with transactions), and the final complexity review—but don't be surprised if you see some integrating questions, building on material from earlier in the term.

4. The final TA-led quiz review will be **Monday, May 13,** during normal class hours (from **2–3pm** in **34-101**). This will be one hour shorter than the other quiz reviews, so we're going to structure it differently. Before coming, please read the on-line lecture notes to get a general overview of the topics on the quiz, and *prepare questions* for us. We will be answering your questions and going over some of these sample problems.

5. Yes, the quiz is open-book, open-notes.

Below are the quizzes from the previous two years. Solutions to these questions will be handed out next week so you can have some time to work on the questions on your own. Have fun and good luck.

Oh, and in case you need this but still don't know what the Web is after the other two quiz handouts (haha), here are the official TA hours:

Dawson Engler	R11–1	NE43-520
Jack Fu	M1–2, R12–1	24-321
Eddie Kohler	M12–1, R3–4	NE43-521a
Ulana Legedza	F3–4, M4–5	NE43-536
Jonathan Litt	M4–5, R4–5	24-321
Yoav Yerushalmi	M4–5:30	24-323

Table of Contents

1. Spring 1995 Quiz 3.
2. Spring 1994 practice questions.

1. *(20 points)* You are designing a computer system to control the power grid for the Northeastern US. If your system goes down, the lights go out and civil disorder —riots, looting, fires, etc.—will ensue. Thus, you have set a goal of having a system MTTF of at least 100 years (about 10^6 hours).

1a. For hardware you are constrained to use a building block computer that has a MTTF of 1000 hours and a MTTR of 1 hour. Assuming that the building block is self checking (i.e., fail stop), how can you arrange to meet your goal? (Briefly explain your answer.)

1b. Your software is 10^6 lines of code. Even after rigorous testing, the software can be expected to have about one bug per 10^3 lines or about 10^3 bugs. You may assume that these bugs are unrelated. Discuss briefly a strategy for how to arrange your software to meet the goal of a MTTF of at least 100 years.

2. *(30 points)* The registrar's office is upgrading its scheduling program for limited-enrollment humanities subjects to make it a network service that anyone can use by RPC. But there is a concern that having multiple threads trying to update the database at the same time could possibly cause trouble.

The program operates as follows; the interface originally had just two entries:

```
Status = Register(subject-name)
```

and

```
Drop(subject-name)
```

where subject-name is a string such as "21W471". The Register procedure checks to see if there is any space left in the subject, and if there is, it increments the class size by one and returns the status value zero. If there is no space, it does not change the class size; instead it returns the status value -1. (This is a very primitive registration system—it just keeps counts!)

As part of the upgrade, two new procedures have been added:

```
Seize(subject-name)
```

and

```
Release(subject-name)
```

Seize sets a lock on the given subject, waiting if necessary for the lock to be released.
Release releases the lock on the given subject.
Here is a typical application program, which registers the caller for two subjects, H1 and H2:

```
integer procedure Register-two(H1, H2)
        status = Register(H1)
        if (status== 0) then
                status = Register(H2)
                if (status == -1)
                        Drop(H1)
                endif
        endif
        return status
```

2a. The goal is that the entire procedure Register-two should be an atomic transaction. Add calls for Seize and Release to the Register-two procedure that obey the Simple Locking Rule (i.e., Seizes before the first read/write, and Releases after the last read/write).

2b. Add calls for Seize and Release that obey the Two-phase Locking Rule, and in addition postpone all Seizes as late as possible and do all Releases as early as possible.

2c. Louis Reasoner has come up with a suggestion that he thinks could simplify the job of programmers creating atomic application programs such as Register-two. His idea is to revise the two programs Register and Drop by moving the Seize and Release statements into them. That is, the procedure:

```
integer procedure Register(subject-name)
        { current code }
        return status
```

would become:

```
integer procedure Register(subject-name)
        Seize(subject-name)
        { current code }
        Release(subject-name)
        return status
```

As usual, Louis has misunderstood part of the problem. Give a brief argument of what is wrong with this idea.

3. *(20 points)* Consider a database system in which all transactions are atomic with respect to system failures and other, concurrent, transactions. Locks on data records along with a log in nonvolatile memory are used to provide these properties. The database itself is stored on disk and cached in main memory by the cache manager. The database system is able to write the following records to the log:

   - BEGIN N: start of transaction number N
   - COMMIT N: commit of transaction N
   - ABORT N: abort of transaction N
   - UNDO N: a record to be able to undo part of transaction N
   - REDO N: a record to be able to redo part of transaction N

   In addition, the system writes DATA records to the on-disk database.

3a. Suppose the cache manager implements a write-through policy. That is, whenever a transaction writes a record to disk to update the database, the record is synchronously forced to disk. Explain what minimum records need to be written to the database and the log, and in what order for the following database operations: start transaction, abort transaction, commit transaction, read of variable, and write of a variable?

Operation	**Write the following record(s) to the log**	**Write the following record(s) to the database**	**Write log or database first?**
Start transaction			
Abort transaction			
Commit transaction			
Read variable			
Write variable			

3b. In the system from part (a), how is the state of the system recovered from the log after a system failure (i.e., contents of main memory are lost)?

4. *(30 points)* Ben Bitdiddle has proposed the simple neat and robust file system (SNARFS). Ben's system has no on-disk data structures other than the disk blocks themselves. Each 4K-byte disk block starts with the following 24 bytes of information:

   - FID (File-ID): a 64-bit number that uniquely defines a file. A FID of zero implies that the disk block is free.
   - SN (Sequence Number): a 64-bit number that identifies which block of a file this disk block contains.
   - T (Time): The time of last update.

   In addition, the first block of a file contains the file name (string), version number, and the FID of its parent directory. The rest of the first block is filled with data. Setting the directory FID to zero marks the entire file free.

   Directories are just files. Each directory need contain only the FID of its parent directory. However as a 'hint' directories may also include a table giving the mapping from name to FID and the mapping from FID to blocks for some of the files in the directory.

   To allow fast access, three in-memory (virtual memory) structures are created each time the system is booted:

   - MAP: an in-memory hash-table that associates a (FID,SN) pair with the disk block containing that block of that file.
   - FREE: a "buddy-system" free list that represents all of the free blocks on disk in a compact manner. (A buddy-system maintains a list of contiguous regions of free disk space.)
   - RECYCLE: a list of blocks that are available for reuse but have not yet been written with an FID of 0.

4a. What is the minimum number of disk I/Os required in Ben's SNARFS system to create a file in an existing directory, and write a 2K-byte string into the file? If the system crashes (i.e., the contents of virtual memory are lost) after these I/Os are completed, the file should be present in the appropriate directory after recovery. Each read or write of a disk block results in one disk I/O. (Briefly explain your answer.)

4b. Ben argues that the in-memory structures can be rebuilt after a crash. Explain briefly what actions are required to rebuild MAP, FREE, and RECYCLE at boot time.

4c. Ben also argues that with appropriate free list management, the write performance of SNARFS can be as fast as for a log-structured file system. How can this be?

Problem 1. *(20 points)* You are examining a 3.5-inch magnetic disk that spins at 7500 RPM, with an average seek time of 6 milliseconds and a data transfer rate of 2 Megabytes/second.

a. How long does it take to read a block of 4 Kilobytes chosen at random?

b. Give the formula that tells the average time to read a 4 Kilobyte disk block, assuming that the hit-ratio of the disk block cache is h.

c. Ben Bitdiddle's workstation has 24 Megabytes of RAM. To increase h, Ben reconfigures the disk block cache to be much larger than one Megabyte. To his surprise he discovers that many of his applications now run slower rather than faster. What has Ben probably overlooked?

d. Louis Reasoner has gone over to the Lab for Computer Science and has disassembled your disk unit to see how it works. Remembering that the centrifuge in the biology lab runs at 36,000 RPM, he has come up with a bright idea on how to reduce the rotational latency of the disk. He suggests speeding it up to 96,000 revolutions per minute. He calculates that the rotation time will now be 625 microseconds.

One of the other UROP's in the lab says this idea is crazy. Explain the line of reasoning that the other student probably has in mind.

Problem 2. (18 points)

Your company has asked you to take the responsibility for developing a storage system for a very demanding application, so you have decided to use a triplexed stable-storage scheme. You will use three ordinary disk drives D1, D2, and D3, and arrange that D2 and D3 store identical copies of each block stored on D1. The disk drives are of a very simple design that does not report read errors. That is, they just return data, whether or not it is valid.

a. You initially construct the application so that it writes a block of data to all three drives concurrently. After a crash occurs during the middle of a write, you are unable to reconstruct the correct data for that block. What is the problem?

b. Describe a simple modification to solve this problem.

c. One day there is a really awful power glitch that crashes all three disks in such a way that each disk corrupts one random track. Fortunately, the system wasn't writing any data at the time. Describe a procedure for reconstructing the data. Explain any cases that your procedure cannot handle.

3. *(20 points)* A system that implements atomic operations writes the following types of records to a log maintained on stable storage:

<AO_i ***start***>	Atomic operation *i* starts.
<AO_i, *x, old, new*>	Atomic operation *i* writes the value *new* over the value *old* for the variable named *x.*
<AO_i ***commit***>	Atomic operation i commits.
<AO_i ***abort***>	Atomic operation i aborts.
<AO_i, AO_j, … AO_k; ***checkpoint***>	At this instant, atomic operations i, j,…k are uncommitted.

Atomic operations start in numerical order. A crash occurs, and the recovery procedure scans the log backwards to the latest checkpoint, finding the following records:

<AO_7, AO_{41}, AO_{42}; ***checkpoint***>
<AO_{43} ***start***>
<AO_{44} ***start***>
<AO_{43}, y, 5, 6>
<AO_{43}, x, 5, 9>
<AO_{43}, ***commit***>
<AO_{44}, y, 6, 4>
<AO_{45} ***start***>
<AO_{45}, z, 3, 4>
<AO_7 ***abort***>
<AO_{41}, q, 1, 9>
<AO_{46} ***start***>
<AO_{45}, y, 4, 3>
<AO_{44} ***commit***>
<AO_{45}, y, 3, 7>
<AO_{41} ***commit***>
<AO_{47} ***start***>
<AO_{46} x, 9, 2>
<AO_{46} w, 0, 1>
<AO_{46} ***commit***>
<AO_{47}, u, 2, 1>
(***** crash happened here *****)

a. How much farther back in the log should the recovery procedure scan?

b. Which atomic operations mentioned in this part of the log are *winners* and which are *losers*?

c. Is it possible to make the list of losers complete with just the information given? Yes/No

d. What are the values of x and y immediately after the recovery procedure finishes? Why?

4. *(5 points)* The log of problem 3 contains (ambiguous) evidence that someone didn't follow coordination rules. What is the evidence?

5. *(5 points)* The intentions-list logging protocol requires that the commit (or abort) record should be written to the log BEFORE any updates to cell storage are made. Identify the best reason for this requirement.

_____ 1. So that the recovery manager knows what to undo.
_____ 2. So that the recovery manager knows what to redo.
_____ 3. Because the log is less likely to fail than the cell storage.
_____ 4. To minimize the number of disk seeks required.

6. *(12 points)* Brooks describes several interesting effects, three of which are listed below:

a. The mythical man-month.
b. The second-system effect.
c. Plan to throw one away.

Following are several root causes. Indicate beside each one the letter of the one effect it is most likely to cause.

_____1. training time
_____2. the first system is likely to be quite spartan
_____3. additional communication
_____4. a second system is likely to have better modularity
_____5. repartitioning effort
_____6. each idea looks good in isolation

Some of these causes can underlie several effects; the "most likely" ambiguity can be resolved by noting that Brooks identifies certain causes as the principal ones.

7. Pat, Diane, and Quincy are having trouble using e-mail to schedule meetings of their 6.033 case study group. Pat suggests that they take inspiration from the 2-phase commit protocol.

(15 points) Which of the following most closely resembles 2-phase commit? (Circle the appropriate Roman numeral.)

```
 I. a. Pat requests everyone's schedule openings.
    b. Everyone replies with a list but does not guarantee to hold all the
          times available.
    c. Pat inspects the lists and looks for an open time.
       If there is a time,
          Pat chooses a meeting time and sends it to everyone.
       Otherwise
          Pat sends a message canceling the meeting.

II. a-c, as in protocol I.
    d. Everyone, if they received the second message,
          acknowledges receipt.
       Otherwise
          sends a message to Pat asking what happened.

III. a-c, as in protocol I.
    d. Everyone, if their calendar is still open at the chosen time
          Send Pat an acknowledgement.
       Otherwise
          Send Pat apologies.
    e. Pat collects the acknowledgements. If all are positive
          Send a message to everyone saying the meeting is ON.
       Otherwise
          Send a message to everyone saying the meeting is OFF.
    f. Everyone, if they received the ON/OFF message,
          acknowledges receipt.
       Otherwise
          sends a message to Pat asking what happened.

 IV. a-f, as in protocol III.
    g. Pat sends a message telling everyone that everyone has confirmed.
    h. Everyone acknowledges the confirmation.
```

(5 points) Below, circle the letter of the step that commits the meeting time.

a b c d e f g h