DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033—Computer System Engineering Handout 31—May 11, 1995

6.033 Quiz #3--When, Where, and How to Prepare

The third 6.033 Quiz will be held on Wednesday, May 17, from 2:05pm to 2:55 p.m. in Walker Memorial.

The quiz will be OPEN BOOK.

Sample Quiz Questions

To give you an idea what you are up against, attached are some sample quiz questions on these same topics from old 6.033 quizzes. The emphasis in different years is different, so you may find that some of the questions seem foreign. You should take the samples as indications of the style of questions asked on these topics, rather than of the material you are expected to know. Beware that most of these questions come from closed-book quizzes.

The answers that were considered appropriate at the time the quizzes were originally given will be in a handout distributed in recitation on Tuesday, May 16.

QUIZ #3. from Spring 1994 2–3 p.m., rooms 34-101, 36-144, 36-153, and 36-156

This quiz contains **seven** problems whose relative weights are indicated by the number of points each is assigned. Please place each answer on the page with the question and hand in this booklet at the end of the examination.

As usual, beware: some parts of some problems are much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. Write down any assumptions and show your work, or you risk losing partial credit. Be neat. If we can't figure out your answer we can't give you credit. For problems that require a numerical answer, be sure to indicate the units of your answer.

Under penalty of major embarrassment if you don't, circle your recitation instance and write your name at the bottom of the page!

1. 10:00—Tennenhouse/McDonald
2. 11:00—Tennenhouse/DeHon
5. 11:00—Corbató/McDonald
7. 11:00—Waldspurger/Manning
6. 12:00—Waldspurger/DeHon
3. 1:00—Kaashoek/Manning
8. 1:00—Corbató/Joseph
4. 2:00—Kaashoek/Joseph

For Official Use Only

Problem	Your Score	Maximum Score
1		20
2		18
3		20
4-5-6		22
7		20
Total		100

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?

Suppose that the operating system maintains a one-Megabyte cache in RAM that hold disk blocks. The access time to this cache is 125 nanoseconds and the data transfer rate is 20 Megabytes/second.

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*?

winners: 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?

x = Because:

```
y =        Because:
```

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

```
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

Section 7: Serializability and Locking (24 pts)

This question uses the following notation:

$R_i(z)$: Thread T_i atomically reads the value of z.

$W_i(z, v)$: Thread T_i atomically writes v as the new value of z.

Consider the following two threads that access two shared variables, x and y, whose values are both initially 0:

T_1: t := $R_1(x)$; $W_1(y, t+1)$
T_2: $W_2(y, 2)$; $W_2(x, 2)$; $R_2(y)$

Part I: Serializability

When T_1 executes to completion before T_2 starts, the sequence of operations on the shared variables is

$R_1(x)$; $W_1(y, 1)$; $W_2(y, 2)$; $W_2(x, 2)$; $R_2(y)$.

T_1 reads x = 0, and T_2 reads y = 2.
The final values of x and y are x = 2 and y = 2.

When T_2 executes to completion before T_1 starts, the sequence of operations is

$W_2(y, 2)$; $W_2(x, 2)$; $R_2(y)$; $R_1(x)$; $W_1(y, 3)$.

T_1 reads x = 2, and T_2 reads y = 2.
The final values of x and y are x = 2 and y = 3.

Consider the following list of some possible concurrent execution sequences:

1. $R_1(x)$; $W_2(y, 2)$; $W_2(x, 2)$; $W_1(y, 1)$; $R_2(y)$
2. $R_1(x)$; $W_2(y, 2)$; $W_2(x, 2)$; $R_2(y)$; $W_1(y, 1)$
3. $W_2(y, 2)$; $W_2(x, 2)$; $R_1(x)$; $R_2(y)$; $W_1(y, 3)$
4. $W_2(y, 2)$; $R_1(x)$; $W_1(y, 1)$; $W_2(x, 2)$; $R_2(y)$

A. (2 pts) Identify an execution sequence in the list above that is serializable.

B. (2 pts) What are the values of x and y, as read by T_1 and T_2 in the execution that you have identified in Part A?

T_1 reads x =
T_2 reads y =

C. (2 pts) What are the final values of x and y after the execution you have identified in Part A?
x =
y =

D. (3 pts) Why is the execution you have identified in Part A serializable?

E. (5 pts) Identify all the executions from the list above that are not serializable. Explain briefly why they are not serializable.

Part II: Locking

The following notation is used to add lock and unlock operations to the thread descriptions of Part I.

$L_i(z)$: Thread T_i acquires exclusive access to z.
$U_i(z)$: Thread T_i releases its lock on z.

Consider the following descriptions of T_1 and T_2, which have been revised to incorporate lock/unlock operations:

1. T_1: $L_1(x)$; t := $R_1(x)$; $U_1(x)$; $L_1(y)$; $W_1(y, t+1)$; $U_1(y)$
   T_2: $L_2(y)$; $L_2(x)$; $W_2(y, 2)$; $W_2(x, 2)$; $R_2(y)$; $U_2(x)$; $U_2(y)$

2. T_1: $L_1(x)$; t := $R_1(x)$; $L_1(y)$; $W_1(y, t+1)$; $U_1(x)$; $U_1(y)$
   T_2: $L_2(y)$; $W_2(y, 2)$; $L_2(x)$; $W_2(x, 2)$; $U_2(x)$; $R_2(y)$; $U_2(y)$

3. T_1: $L_1(x)$; t := $R_1(x)$; $L_1(y)$; $U_1(x)$; $W_1(y, t+1)$; $U_1(y)$
   T_2: $L_2(x)$; $L_2(y)$; $W_2(y, 2)$; $W_2(x, 2)$; $U_2(x)$; $R_2(y)$; $U_2(y)$

4. T_1: $L_1(x)$; t := $R_1(x)$; $L_1(y)$; $W_1(y, t+1)$; $U_1(x)$; $U_1(y)$
   T_2: $L_2(y)$; $L_2(x)$; $W_2(y, 2)$; $W_2(x, 2)$; $U_2(y)$; $R_2(y)$; $U_2(x)$

F. (5 pts) Identify all of the entries in the above list in which both T_1 and T_2 obey the two-phase locking protocol. Explain briefly.

G. (5 pts) Identify one entry from the list which can lead to deadlock. Explain briefly.

Section 8: Crash Recovery (18 pts)

A transaction system writes the following types of records to a log maintained on stable storage:

< T_i start >: Transaction T_i starts
< T_i, x, *old*, *new* >: Transaction T_i writes the new value, *new*, over the old value, *old*, for x.
< T_i commit >: Transaction T_i commits.

The system design is based on a write-ahead log protocol. During recovery from the crash, the transaction manager finds the following records in the log.

beginning of log
< T_1 start >
< T_2 start >
< T_1, y, 0, 1 >
< T_1, x, 0, 1 >
< T_1 commit >
< T_2, y, 1, 2 >
< T_3 start>
< T_3, z, 3, 4 >
< T_4 start >
< T_3, y, 2, 3 >
< T_2 commit >
< T_3, y, 3, 4 >
< T_5 start >
< T_4, x, 1, 2 >
< T_4, w, 0, 1 >
< T_4 commit >
< T_5, u, 2, 1 >
< T_5, w, 1, 6 >
end of log

A. (6 pts) Which transactions should persist after crash recovery?

B. (6 pts) What are the values of x and y immediately after crash recovery processing (i.e. before any new transactions are started)? Why?

C. (6 pts) The “write ahead log rule” specifies that an update record must be written to the log before the corresponding update to the data on disk. Explain briefly why this is necessary.

7.3 (Suggested by S. Heller) Find the one best choice in each of the following:

A. This property of atomic actions allows them to cope with failures:

1. Constituent instructions of atomic actions are elementary operations.
2. Constituent instructions of atomic actions are atomic.
3. Atomic actions can back up and appear as if never started.
4. Atomic actions are carefully defined so that they can be proven correct.

B. This property of atomic actions allows them to deal properly with concurrency:

1. All results could have been achieved by some serialization of the actions.
2. All atomic actions complete in one step (the smallest identifiable time unit).
3. Atomic actions can occur simultaneously.
4. Any set of atomic actions that overlap in time either all commit or all abort.

[7831]

a) In the context of nested atomic transactions, in what situation is compensation required? (Note: compensation is undoing the effect of a committed atomic operation.)

b) Under what (two) conditions is compensation possible?

7.34 (Suggested by R. H. Halstead, Jr.) In the Stability, Inc., data storage system, when a crash occurs, all programs currently running are stopped, and all program state information is lost. Only data kept in stable storage (the database and any logs kept) are preserved across a crash. A database system that is built on this data storage system executes atomic transactions that read and write multiple locations in the database.

A. Jim proposes to ensure failure atomicity in the database system by keeping a log that records every value written into the database (but does not record any other actions). The log record is written to stable storage before the database write itself is performed. Each log entry includes the address written into and the new data written here. Is this log, plus a copy of the database at the time of a crash, sufficient to assure failure atomicity across the crash?

B. Jean claims that Jim is all wrong, what is really needed is a log just like the log described by Jim, but which records the old contents of each location just before it was written, rather than the new information written there. Is Jean's log, plus a copy of the database at the time of a crash, sufficient to assure failure atomicity across the crash?

C. Jules asserts that failure atomicity can be assured simply by an appropriate scheme for locking between transactions, without using a log at all. Is Jules right?

[8133]

7.9 (Suggested by C. Leiserson) This problem explores the implementation of atomic storage--storage for which a multiword put operation has one of two outcomes.

a. What are the two outcomes?

Assuming that careful storage [which guarantees that if its operations terminate, they were successful] is implemented by careful-get and careful-put, we program atomic-get and atomic-put using version-numbered duplicate copies as in the SABRE system. (V_1 and V_2 are two copies of a version number, while D_1 and D_2 are two copies of the data.)

atomic-get: [careful-get (V_1, D_1, V_2, D_2);
    return {if $V_1 = V_2$ then D_1 else D_2}]

1. atomic put: [careful-get (V_1, D_1, V_2, D_2);
2. if $V_1 = V_2$ then {if D_1 != D_2, careful-put D_1 onto D_2}
3. else {careful-put D_2 onto D_1;
4. careful-put V_2 onto V_1};
5. new-V = V_1 + 1;
6. careful-put new-V onto V_1;
7. careful-put new-data onto D_1;
8. careful-put new-V onto V_2;
9. careful-put new-data onto D_2;
10. return;]

b. Where is the commit point of atomic-put?

c. Can step 9 be left out of the programs? Explain why or why not.

[8232]

7.10 (JHS) Rapidata Corporation markets a data management system that is intended to allow transactions to proceed with a maximum of concurrency. The system asks that the programmer of a procedure to be executed as an atomic action follow only these two (unfortunately, incorrect) rules:

rule 1: the procedure may read only data that was written by previously committed such procedures.

rule 2: the procedure must mark any data item it will write, and no such procedure may write into a data item already marked by another such procedure.

(When such a procedure commits, all its marks vanish.)

Now consider the following two procedures, noting that both carefully follow the rules:

aa1:	mark (y);	aa2:	mark (x)
	buffer = get(x);		value = get(y);
	put(y) = buffer;		put(x) = value;

a) Suppose these two procedures execute concurrently, and happen to do their steps in the following order:

time	1.	aa1: mark(y);	.
	2.	buffer = get(x);	.
	3.	.	aa2: mark(x);
	4.	.	value = get(y);
	5.	.	put(x) = value;
	6.	put(y) = buffer;	.

A skeptical 6.033 classmate named Tracy Swallow claims that this example shows that the coordination rules are not adequate. To illustrate, Tracy chooses starting value x = 3 and y = 4. At the end, y = 3 and x = 4.

Tracy claims that this result is not correct and that aa1 and aa2 are therefore not genuine atomic actions. What does Tracy mean by “not correct”?

b) For starting values x = 3 and y = 4 what are all the possible answers that Tracy would consider correct?

[8233]

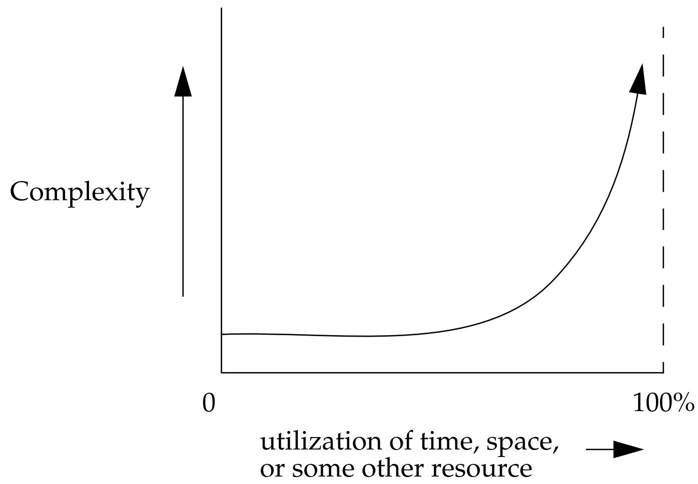
9.4 (Suggested by H. Forsdick) A contract software production firm, the NJU Company, has just finished an extended programming project that was estimated to cost $500,000 and take 3 months using 50 programmers. In actuality, the project lasted 12 months, involved at various times 100 different programmers and cost $2,500,000. When it was finished the system constructed did not meet the original specifications. The client made the following statement at the end of the project:

We are very dissatisfied. Even before we spent any money on software, we had spent $1,000,000 on the hardware of this system. That was already stretching the limits of our financial resources. We told the NJU Company that we need this system quickly to demonstrate to our board of directors that the computer purchase would help our company. Also, we found the NJU people to be extremely uncooperative to our suggestions for small changes to our original proposal. After all, they told us that they would have plenty of people working on this project and certainly some of them could be used to handle matters like changes to the system. Finally, as the last straw, when we were given the system, we had to do some of the work on the system ourselves. Our competent chief systems programmer had to spend a long time wading through a bunch of flow charts to find out where changes should go and then spent forever debugging the changes because the NJU programmers forgot to save some processor registers on certain procedure calls.

What did NJU do wrong in this system? Specifically, point out four problems and solutions to those problems as suggested by Brooks in *The Mythical Man Month*.

[7554]

9.5 (JHS) Consider the following general graph:



Give as many different examples as you can of situations in computer systems engineering that illustrate this phenomenon.
[7833]

9.6 (Developed by several of the 1981 teaching staff) Below is a list of eight goals and fourteen techniques (i.e., concepts or mechanisms that aid in achieving one of the goals). For each technique, indicate which goal is most greatly aided by that technique.

GOALS

A) understanding complex systems
B) easy transfer of rights to access information
C) easy to tell who can and who cannot access information
D) easy sharing of data and procedures
E) detecting errors
F) matching the demand for a device to its delay and bandwidth characteristics
G) creating atomic operations
H) avoiding deadlock
I) minimizing both cost and delay of a resource

TECHNIQUES

1. ___ locking resources in ascending order
2. ___ system objectives
3. ___ store and forward packet switching
4. ___ capabilities
5. ___ transaction logs
6. ___ layering of protocols
7. ___ user-dependent bindings
8. ___ passwords on files
9. ___ two-phase locking
10. ___ closures
11. ___ combining high-performance local and medium-performance long-distance network technologies
12. ___ positive acknowledgement of packets in a network protocol
13. ___ multi-level memory
14. ___ access control lists

[8135]