DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033—Computer System Engineering   Handout 47: May 15, 1997

Quiz 3 Solutions

All 16 problems on this exam are short answer questions; each requires a choice, a Y or N, a number, or a brief explanation in the space provided. In order to receive credit you must fill in the blank(s) or mark the correct answer for each question.

Put your name on this cover sheet AND at the bottom of each page of this booklet. Be sure that you don't put part of the answer to a problem on the back of a sheet for another problem.

Some problems may be much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find problems ambiguous, be sure to write down any assumptions you make. Be neat. If we can't figure out your answer we can't give you credit. On answers that involve numbers, be sure to clearly specify the units of your answer.

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.

Circle your recitation:

10:00	1. Lynch/Banerji		
11:00	2. Lynch/Lau	7. Engler/Winship	12. Shrira/Banerji
12:00	6. Engler/Lau		
1:00	3. Sussman/Sapuntzakis	9. Ward/MacGregor	
	8. Saltzer/Schindler	11. Shrira/Winship	
2:00	4. Sussman/MacGregor	5. Ward/Schindler	
	10. Saltzer/Sapuntzakis		

Name: ________________________

Nontrivial Pursuits...

1. [8 points]: RAID 5 differs from RAID 4 in that parity is spread among drives, with the advantage that

A. Single-drive failures are tolerated.
B. Read and write performance in the absence of errors is enhanced.
C. Write performance in the absence of errors is enhanced.
D. Locating data on the drives is easier.
E. There's no real advantage, its just another stupid fad.

Although the big improvement is in write performance, there is also a small improvement in read performance. Since RAID 5 places useful data on every disk, there is more opportunity for parallel reads.

(Circle letter to left of best answer)

2. [8 points]: A certain disk drive takes an average of 10 ms to move its head to an arbitrary track, rotates one full revolution every 8 ms, and has 10 sectors per track. What is the average time necessary for an arbitrary 1-sector read?

A. 10 ms.
B. 10.8 ms.
C. 14 ms.
D. 14.8 ms.
E. 18 ms.
F. 18.8 ms

(Circle letter to left of best answer)

3. [8 points]: Two-phase locking within transactions assures that

A. No deadlocks will occur.
B. Results will correspond to some serial execution of the transactions.
C. Resources will be locked for the minimum possible interval.
D. Neither gas nor liquid will escape.
E. Transactions will succeed even if one lock attempt fails.

(Circle letter to left of best answer)

4. [8 points]: The ATM of a dishonest bank in the Cirrus system might cheat by claiming to have dispensed money when in fact it hadn't. Cirrus detects this by

A. Multi-key hyperbolic affine encryption techniques.
B. Smart cards.
C. Batch posting.
D. Four-message protocol.
E. Customer complaints.

(Circle letter to left of best answer)

Name: ________________________

Sick Transit

Gloria Mundi, an MIT alumna who dropped 6.033 immediately following the second quiz, is undertaking to resurrect the failed London Ambulance Service as a new streamlined company called Sick Transit. She has built a new computer she intends to use for processing ST's transactions. Although she never understood the System R paper, she is sure that a much simpler approach can be used for reliable transaction processing.

A key component in Gloria's machine is a highly reliable sequential-access infinite tape, which she plans to use as an **append-only** transaction log. Records can be appended to the tape, but once written are immutable and stable. Records on the tape can be read any number of times, from front-to-back or from back-to-front. There is no disk in the ST system; the tape is the only non-volatile storage.

Because of the high cost of the infinite tape, Gloria compromised on the quality of more conventional components like RAM and CPU, which fail frequently but are fail-fast in the sense that every error causes an immediate system crash. Gloria plans to ensure that, after a crash, a consistent state is recoverable from the log on the infinite tape.

Gloria's code uses transactions, each identified by a unique transaction ID. The visible effect of a completed transaction is confined to changes in global variables, whose WRITE operations are logged. The log will contain entries recording the following operations:

BEGIN(tid) start a new transaction, whose unique ID is *tid*

COMMIT(tid) commit a transaction

ABORT(tid) abort a transaction

WRITE(tid,variable,old-value,new-value)

write a global variable, specifying previous & new values.

To keep the system simple, Gloria plans to use the above forms as the application-code interface to the transaction system, in addition to a *READ(tid,variable)* call which returns the current value of *variable*. Each of the calls will perform the indicated operation and write a log entry as appropriate. Reading an unwritten variable is to return ZERO.

Gloria begins by considering the single-threaded case (only one transaction is active at any time). She stores values of global variables in a symbol table in RAM. Gloria is now trying to figure out how to reset variables to committed values following a crash, using the log tape.

5. [8 points]: In the single-threaded case, what value should variable v be restored to following a crash?

A. 37

B. *new-value* from the last logged *WRITE(tid,v,old-value,new-value)* to v, or ZERO if unwritten.

C. *new-value* from the last logged *WRITE(tid,v,old-value,new-value)* not followed by an *ABORT(tid)*, or ZERO if unwritten.

D. *new-value* from the last logged *WRITE(tid,v,old-value,new-value)* followed by a *COMMIT(tid)*, or ZERO if unwritten.

E. Either *old-value* or *new-value* from the last logged *WRITE(tid,v,old-value,new-value)* to *v*, depending on whether that *WRITE* is followed by a *COMMIT* on the same *tid*, or ZERO if unwritten.

(Circle letter to left of best answer)

Gloria now tries running concurrent transactions on her system. Of course, accesses to the log continue to be serialized by the sequential-access tape drive.

Her first trial involves parallel execution of the transactions:

```
BEGIN(t1);                         BEGIN(t2);
{  int t1x=READ(x);                {  int t2x=READ(x);
   int t1y=READ(y);                   int t2y=READ(y);
   WRITE(t1,x,t1x,t1y+1);             WRITE(t2,y,t2y,t2x+2);
   COMMIT(t1);                        COMMIT(t2);
}                                  }
```

Note that the initial values of x and y are ZERO, as are all uninitialized variables in her system. Here the READ primitive simply returns the most recently written value of a variable from the RAM table, ignoring commits.

6. [8 points]: In the absence of locks or other synchronization mechanism, will the result necessarily correspond to some serial execution of atomic transactions?

A. Yes.

B. No, since the execution might result in x=3, y=3.

C. No, since the execution might result in x=1, y=3.

D. No, since the execution might result in x=1, y=2.

(Circle letter to left of best answer)

Gloria is considering using locks, and automatically adding code to each transaction so as to guarantee serial consistency of transactions. She would like to maximize concurrency; she is, however anxious to avoid deadlocks. For each of the following proposals, identify whether the approach (1) yields semantics consistent with some sequential execution of atomic transactions and (2) introduces potential deadlocks.

7. [4 points]: A single, global lock which is locked at the start of each transaction and released at COMMIT.

Transaction Atomicity (Y/N)? __Y__; Introduces potential Deadlocks (Y/N)? _N__

8. [4 points]: A lock for each variable. Every READ or WRITE operation is immediately surrounded by a LOCK/RELEASE of that variable's lock.

Transaction Atomicity (Y/N)? __N__; Introduces potential Deadlocks (Y/N)? _N__

9. [4 points]: A lock for each variable. Every variable READ or WRITTEN within a transaction is locked immediately prior to the first reference to that variable in the transaction; all locks are released at COMMIT.

Transaction Atomicity (Y/N)? __Y__; Introduces potential Deadlocks (Y/N)? _Y__

10. [4 points]: A lock for each variable. All variables READ or WRITTEN within a transaction are locked, in alphabetical order, immediately following the BEGIN. They are all released at COMMIT.

Transaction Atomicity (Y/N)? __Y__; Introduces potential Deadlocks (Y/N)? _N__

11. [4 points]: A lock for each variable. All variables WRITTEN within a transaction are locked, in alphabetical order, immediately following the BEGIN. They are all released at COMMIT.

Transaction Atomicity (Y/N)? __N__; Introduces potential Deadlocks (Y/N)? _N__

In the general case (concurrent transactions) Gloria would like to avoid having to read the entire log during crash recovery. She proposes periodically adding a CHECKPOINT entry to the log, and reading the log backwards from the end restoring committed values to RAM. The backwards scan should end as soon as committed values have been restored to all variables. Each CHECKPOINT entry in the log contains current values of all variables and a list of open (uncommitted) transactions at the time of the CHECKPOINT.

12. [10 points]: What portion of the tape must be read to properly restore values committed at the time of the crash?

A. All of the tape; checkpoints don't help.

B. Enough to include the BEGIN from each transaction which was uncommitted at the time of the crash.

C. Enough to include the last CHECKPOINT, as well as the BEGIN from each transaction which was uncommitted at the time of the crash.

D. Enough to include the last CHECKPOINT, as well as the BEGIN from each transaction which was uncommitted at the time of the last checkpoint.

The checkpointed values may include the results of actions of a transaction that was in progress at the checkpoint but aborted later. (Note: there is an ambiguity in the problem statement that can allow one to conclude that answer C is correct.)

(Circle letter to left of best answer)

Simplicity Winns, Gloria's onetime MIT classmate, observes that since global variable values are recoverable from the log their storage in RAM is redundant. She proposes eliminating the RAM as well as all of Gloria's proposed locks, and implementing a ***READ(tid,var)*** primitive which returns an appropriate value of ***var*** by examining the log. Note that, in the following description, the arguments ***tid*** and ***var*** to the ***READ*** call are in bold type.

Simplicity's plan is to implement ***READ*** so that each transaction *t* "sees" the values of global values at the time of *BEGIN(t)*, as well as changes made within *t*. She quickly sketches an implementation of ***READ*** which she claims gives appropriate transaction atomicity semantics:

1. Set variables *winners* to empty list and *prior* to FALSE;
2. Loop over each entry *E* in log, starting at last and working backward:
   - 2.1: if E is *BEGIN(**tid**)* then set *prior* to TRUE;
   - 2.2: if E is *COMMIT(etid)* and *prior* is TRUE, add e*tid* to *winners* list;
   - 2.3: if E is *WRITE(etid,**var**,old-val,new-val)* then
     - 2.3.1: if *etid* is ***tid***, then return *new-val*;
     - 2.3.2: if *etid* is on *winners* list, then return *new-val*;
   - 2.4: continue looping to next earlier entry. If list is exhausted, return 0.

Gloria is a little dazed by Simplicity's quick synopsis, but thinks that Simplicity, who aced 6.033, is likely correct. Gloria asks your help in figuring out what Simplicity's algorithm actually does.

13. [10 points]: Suppose transaction t *READ*s variable x but does not write it. Will each READ of x in *t* see the same value? If so, concisely describe the value returned by each *READ*; if not, explain.

Yes. The value returned is the last value of x that was written by any transaction that committed prior to BEGIN(t). (If there is no such value than 0 is returned.)

(Explain Briefly)

14. [8 points]: Does Simplicity's scheme REALLY offer atomic transaction semantics yet avoid deadlocks?

A. Yup. Read it and weep, Gloria.

B. It doesn't introduce deadlocks, but doesn't guarantee serialized atomic transaction semantics either.

C. It gives serialized atomic transactions, but introduces possible deadlocks.

D. Simplicity's approach doesn't work even when there's no concurrency -- it gives wrong values for variables. Maybe she aced 6.033 because her partners on the second project were 6-sigma hackers.

E. Isn't it about time for this quiz to be over?

(Circle letter to left of all applicable answers)

Gift Department

You all realize that the real motivation of the *Sick Transit* problem is a stupid pun. Let's see if you can identify it:

15. [2 points]: What does *sic transit gloria mundi* actually mean?

A. Thus passes a glorious Monday.

B. Thus pass the earthly glories.

C. Gloria threw up on the T Monday.

D. This bus for First Class and Coach.

E. This is the last straw! If I wanted to take @#%!*% *Latin* tests, I'd have gone to Oxford. I'm reporting this course to the committee on curricular improprieties.

(Circle letter to left of best answer)

Finally, you of course recall the sage advice of our guest lecturer, Bob Metcalfe:

16. [2 points]: Metcalfe suggests you start your writing with:

A. "Bob Metcalfe, visionary inventor of the Ethernet, ..."

B. "You"

C. "The World Wide Web is rapidly gaining popularity"

D. "I"

E. "Howdy, pardner."

END OF QUIZ 3

END OF 6.033!

(some day you'll look back on all this fondly...)