Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2015

Quiz 2

There are 21 questions and 11 pages in this quiz booklet. Answer each question according to the instructions given. You have **120 minutes** to answer the questions.

Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

If you find a question ambiguous, be sure to write down any assumptions you make. **Be neat and legible.** If we can't understand your answer, we can't give you credit!

Write your name in the space below. Write your initials at the bottom of each page.

THIS IS AN OPEN BOOK, OPEN NOTES, OPEN LAPTOP QUIZ, BUT DO NOT USE YOUR LAPTOP FOR COMMUNICATION WITH OTHERS. TURN ALL NETWORK/COMMUNICATION DEVICES OFF.

CIRCLE your recitation section:

10:00	1. Dina/Andrew	2. Arvind/Manali	3. Karen/Ellen	
11:00	4. Dina/Andrew	5. Arvind/Manali	6. Karen/Ellen	
12:00	7. Sam/David			
1:00	8. Peter/Cong	9. Sam/David	10. Martin/Webb	13. Mark/Amy
2:00	11. Peter/Cong	12. Mark/Amy	14. Martin/Webb	

Do not write in the boxes below

1-4 (22)	5-6 (10)	7-9 (11)	10-11 (10)	12-13 (10)	14-15 (8)	16-18 (16)	19-21 (13)	Total (100)

Name:

Initials:

I RAID, LFS, GFS, and PNUTS

1. [6 points]: Consider the following statements about a redundant array of inexpensive disks (RAID):
(Circle True or False for each choice.)

(A) True / False A disk array is said to experience a failure if any disk in the array fails. With this definition of failure, the MTTF of a disk array is larger than the MTTF of a single disk in the array. **False - it's smaller to the MTTF of a single disk**

(B) True / False RAID level 5 has lower storage overhead than RAID level 4. **False**

(C) True / False RAID level 5 performs better than RAID level 4 for small, random writes. **True**

2. [6 points]: Consider the following statements about the Log-Structured File System (LFS):
(Circle True or False for each choice.)

(A) True / False LFS selects the oldest (highest age) segment to clean. **False**

(B) True / False Compared to UNIX FFS, LFS improves the performance of small random file writes, but has worse performance on small, random file reads. **False**

(C) True / False For a given, non-negligible rate of file-system operations, the time to recover from a crash is lower if checkpoints are taken more frequently. **True**

3. [6 points]: Circle True or False for each statement below about the Google File System (GFS).

(A) True / False GFS optimizes for writes that append data to a file, rather than writes to random offsets. **True**

(B) True / False GFS uses a larger chunk size than traditional UNIX block sizes. The authors defend this design decision by observing that disks have become much faster since the UNIX file system was designed, and that the chunk size must be proportionally larger. **False**

(C) True / False In GFS, detecting data corruption requires comparing the data stored for the same chunk on different replicas. **False - they use checksums**

4. [4 points]: Yahoo!'s PNUTS system exhibits which of the following design decisions?
(Circle True or False for each choice.)

(A) True / False All writes are sequenced through the master site. **True**

(B) True / False Timeline consistency across multiple records. **False - Timeline consistency per record, not across records**

Initials:

II All-or-nothing atomic sector

We studied the following protocol to implement an all-or-nothing disk sector in 6.033 this term:

```
all_or_nothing_put(s, data):
    status = careful_get(s.D0, buffer)
    if status == OK:
        careful_put(s.D1, data);
        careful_put(s.D0, data);
    else:
        careful_put(s.D0, data);
        careful_put(s.D1, data);
```

```
all_or_nothing_get(s, data):
    # data contains result of get
    status = careful_get(s.D0, data)
    # checksum (OK means not corrupt)
    if status == OK:
        return;
    careful_get(s.D1, data);
```

The failure model is a power failure, hardware crash, or software crash that could corrupt the data being written to a disk sector. As soon as such a failure occurs, the system reboots. There is no decay of data on a disk to worry about. Recall that `careful_put` and `careful_get` use a checksum to verify the integrity of the data written to `D0` and `D1`. Assume they are implemented correctly.

5. [4 points]: Circle **True** or **False** for each statement below.

(A) True / False If, during `all_or_nothing_put`, the first `careful_put` succeeds, and a failure occurs before the second `careful_put` starts, then the subsequent call to `all_or_nothing_get` (assuming no failure during this call) is guaranteed to see the result of the **first** `careful_put`. **False - if status is OK in put, get will see the result in D0, not D1**

(B) True / False If, in `all_or_nothing_get`, we changed `s.D0` to `s.D1` and `s.D1` to `s.D0`, but kept `all_or_nothing_put` unchanged, then it would no longer provide all-or-nothing semantics. **False**

Initially, the disk sectors `D0` and `D1` each store uncorrupted data x. Suppose `all_or_nothing_put` attempts to store a new value y into the sector `s`. Represent the state of the disks (`D0, D1`) as (a, b), where $a = x, y$, or !. Here, x and y are the uncorrupted values stored and "!" represents corrupted data. There are nine possible states (three choices for `D0` and three for `D1`).

6. [6 points]: From state (x, x), what possible next states that the system could be in after the call to `all_or_nothing_put(s, y)`. The call could succeed or it could fail at any time during its execution. **(Circle True or False for each choice.)**

(A) True / False $(!, !)$ **False**

(B) True / False $(!, x)$ **False**

(C) True / False $(!, y)$ **True - failure during second put**

(D) True / False $(x, !)$ **True - failure during first put**

(E) True / False (x, x) **True - failure before first put**

(F) True / False (x, y) **True - failure between puts**

(G) True / False $(y, !)$ **False**

(H) True / False (y, x) **False**

(I) True / False (y, y) **True - no failure**

Initials:

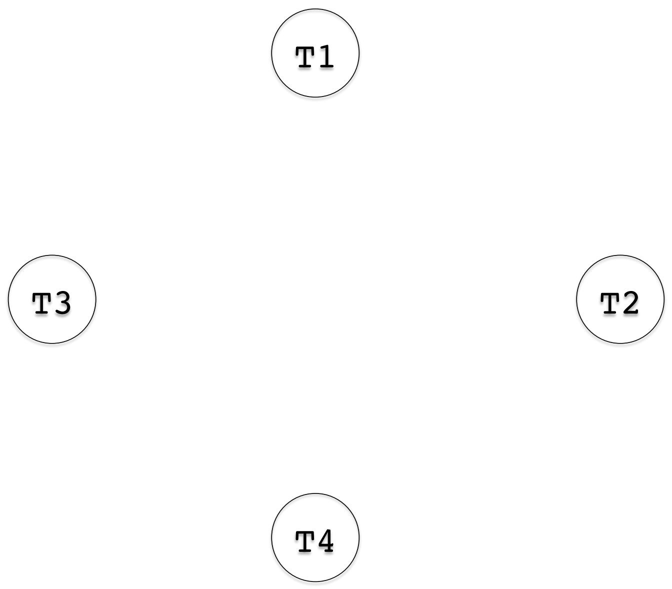
III Serializability in Transactions

A transaction processing system runs the steps of transactions `T1, T2, T3,` and `T4` in the order shown (time goes downwards; step i runs before step $i+1$). All transactions commit after step 8 below.

T1:	**T2:**	**T3:**	**T4:**
		`1. read(z)`	
	`2. read(x)`		
			`3. write(x)`
		`4. write(y)`	
`5. read(y)`			
	`6. write(z)`		
			`7. read(y)`
`8. write(z)`			

7. [5 points]: Draw the precedence graph (conflict graph) corresponding to this schedule of steps.

Correct solution has five edges: T3 $\rightarrow$ T1, T3 $\rightarrow$ T2, T3 $\rightarrow$ T4, T2 $\rightarrow$ T1, T2 $\rightarrow$ T4



8. [3 points]: List **all** the serial transaction orders equivalent to the schedule of steps shown.

T3, T2, T1, T4 and T3, T2, T4, T1

9. [3 points]: True / False The schedule of steps shown is achievable by a transaction processing system using two-phase locking. Assume that a lock on an item is acquired as the first part of the step in which the item is first used. **False - T2 would have to release a lock on x after step 2 so that T4 can acquire it in step 3, but T2 would have to acquire a lock on z in step 6, violating 2PL.**

Initials:

IV Hammurabi's write-ahead log

A team of archeologists has recently deciphered an ancient Sumerian stone tablet from nearly 3800 years ago, determining that it encodes a portion of a write-ahead log that appears to describe an historic employee-salary database! In modern 6.033 notation, the log says the following:

```
 ...    ...
 41.    <CHECKPOINT 100, 101, 102>
 42.    <BEGIN 103>
 43.    <103, Chief_Minister, 50, 75>
 44.    <BEGIN 104>
 45.    <103, Court_Jester, 20, 15>
 46.    <COMMIT 103>
 47.    <104, Chief_Minister, 75, 70>
 48.    <BEGIN 105>
 49.    <105, Chief_Accountant, 30, 300>
 50.    <ABORT 100>
 51.    <101, Prince, 30, 40>
 52.    <BEGIN 106>
 53.    <105, Chief_Minister, 70, 72>
 54.    <COMMIT 101>
 55.    <BEGIN 107>
 56.    <106, Court_Jester, 15, 16>
 57.    <ABORT 106>
/****** CRASH HAPPENED HERE; NO MORE LOG ENTRIES ******/
```

- $\langle$CHECKPOINT $i, j, \ldots\rangle$: This log entry says that transactions $i, j, \ldots$ are active.
- $\langle i, x, old, new\rangle$: transaction i changed variable x (Chief_Minister, Court_Jester, etc.) from old to new.
- ABORT i means that the system successfully aborted transaction i, writing this log entry at the end of the abort procedure.

The Sumerians had not invented the concept of a buffer manager, so all writes were forced immediately to stable cell storage (yep, another stone tablet!). In modern terminology, they used FORCE and STEAL.

10. [6 points]: Which transactions mentioned in this portion of the log must be **undone** and which must be **redone** on crash recovery?

(Circle ALL that apply)

UNDO: `100` `101` `102` `103` `104` `105` `106` NONE

REDO: `100` `101` `102` `103` `104` `105` `106` NONE

Undo - 102, 104, 105. Redo - None.

11. [4 points]: Given this log, does this Sumerian system use two-phase locking to handle concurrent transactions? **Circle** one of these three choices **and** provide a **brief** one-line reason.

YES. **NO.** **WE CANNOT BE CERTAIN.**

Reason: No - Lines 47 and 53 that update Chief_Minister are concurrent writes of the same variable. (Note that the transaction on line 43 has COMMITTED on line 46, and is not concurrent with lines 47 and 53.) ______________________

Initials:

V Distributed transactions

12. [6 points]: Consider the two-phase commit protocol we studied in 6.033 between a coordinator and multiple machines running distributed transactions.

(Circle True or False for each choice.)

(A) **True / False** If a machine has responded to a "prepare to commit" message with "prepared", then it **must** have written its updates to stable storage (disk). **True**

(B) **True / False** The coordinator **must not** abort the transaction after it receives a "prepared" message from some machine. **False**

(C) **True / False** Deadlocks are impossible with a central coordinator running two-phase commit. **False**

13. [4 points]: Alyssa, Ben, Charlie, and Donna are trying to sync up state in a multi-player video game they are playing, using an optimistic replication protocol with vector timestamps. Donna observes the following vector timestamps for different proposed game states from the other three:

- From Alyssa: $\langle 10, 20, 6, 0\rangle$
- From Ben: $\langle 8, 19, 7, 0\rangle$
- From Charlie: $\langle 9, 19, 7, 0\rangle$

Prior to receiving these, she has a version of the game state with vector timestamp $\langle 8, 18, 7, 0\rangle$.

Donna would like to use the latest version that does not conflict with what she already has. Whose game state should Donna use?

(Circle ALL that apply)

(A) Alyssa's.

(B) Ben's.

(C) Charlie's.

(D) Donna's.

Charlie's. Alyssa's conflicts with Donna's, and Charlie's is more recent than Ben's. We gave partial credit for saying (C) and (B), since Ben's does not conflict with Alyssa's.

Initials:

VI Principal Authentication and Buffer Overflows

In this section, and all that follow, "|" implies concatenation.

14. [6 points]: Ben runs a webserver that uses passwords to authenticate its clients. Ben has taken 6.033, and knows that he should use salted hashes to prevent rainbow-table attacks.

Each time a user registers, Ben's server generates a random salt (assume that this process is correct, i.e., there is no issue with his random-number generator, and the salt value is at least 10 digits). Bob knows that he will need to store information related to the username, salt, and password, but he isn't sure how.

Each of the below options give three items that Ben might store on his server. For each of the options, determine whether Ben's system will be resistant to rainbow-table attacks. I.e., whether it will be infeasible for an attacker to guess Alice's password—even if it is a common one such as "123456"—without resorting to a brute force (exhaustive enumeration) attack.

(Circle True or False for each choice.)

(A)
```
username
salt
hash(password) | salt
```

True / False Ben is resistant to rainbow-table attacks. **False**

(B)
```
hash(username)
salt
hash(password | salt)
```

True / False Ben is resistant to rainbow-table attacks. **True**

(C)
```
username
salt
hash(password | hash(salt))
```

True / False Ben is resistant to rainbow-table attacks. **True**

15. [2 points]: Ben decides to hire someone to implement salted hashes for him, and they do so correctly. He moves onto having his users use session cookies. In Ben's scheme a user's cookie is:

{`username, expiration, hash(S | username | expiration)`}

where S is a value known only to the server.

True / False Ben could shorten his cookie scheme to use {`username, expiration, hash(S)`} without introducing any new security vulnerabilities. **False - A user could change the username and log in as someone else, or change the expiration date and log in forever**

Initials:

VII Secure Channels and DNSSEC

16. [6 points]: Alice wants to send messages to Bob via symmetric-key cryptography, but knows that Diffie-Hellman key exchange is open to a man-in-the-middle attack from Eve. To get around that, she decides to use signatures. Alice and Bob do the following:

- Alice generates a key-pair, (PK_A, SK_A). She publishes PK_A using a secure website; assume Bob will learn the correct value for PK_A.
- Alice and Bob use Diffie-Hellman to exchange a symmetric key, k.
- When Alice sends a message m to Bob, she sends:

  $$encrypt(k, m|seq) \mid MAC(k, m|seq) \mid SIGN(SK_A, m|seq)$$

  where seq is a monotonically increasing sequence number (i.e., each value of seq is strictly greater than the one before it).

Suppose that Bob knows PK_A, that the key exchange is complete, and that Bob has just received his first transmission from Alice, containing a message m.

(Circle True or False for each choice.)

(A) **True / False** Bob can guarantee that Eve does not know the value of m (i.e., their scheme provides confidentiality). **False - The Diffie-Hellman MITM still exists here.**

(B) **True / False** Bob can guarantee that the transmission originated from Alice (i.e., their scheme provides authenticity). **True**

(C) **True / False** Bob can guarantee that Eve did not tamper with the transmission (i.e., their scheme provides integrity). **True**

17. [2 points]: Consider DNSSEC, as described in "Security Vulnerabilities in DNS and DNSSEC" by Ariyapperuma and Mitchell.

True / False DNSSEC distributes keys using a centralized infrastructure, similar to the certificate authorities discussed in class. **False - key exchange is distributed in DNSSEC**

Initials:

18. [8 points]: Alice and Bob are communicating via two symmetric keys, k_1 and k_2 (assume that no one else in the network knows the values of k_1 and k_2). They use the following protocol:

- When Alice wants to send a message m to Bob, she sends

$$encrypt(k_1, m|seq_A) \mid MAC(k_1, m|seq_A)$$

  where seq_A is a monotonically increasing sequence number. Bob will ignore Alice's transmission if it contains a sequence number that he has already received from her.

- When Bob wants to communicate with Alice, he does the same, but uses k_2 instead of k_1, and a different (but still monotonically increasing) sequence number:

$$encrypt(k_2, m|seq_B) \mid MAC(k_2, m|seq_B)$$

  Similarly, Alice will ignore any transmissions from Bob that contains a sequence number she has already received from him.

Alice and Bob would like to take some shortcuts in their protocol.

(Circle True or False for each choice.)

(**A**) Instead of generating two keys, they use a single key, k. However, they change their sequence number scheme: Bob will ignore any transmission from Alice with a sequence number he has already seen *and also* any transmission with an **even** sequence number. Alice will ignore any transmission from Bob with a sequence number she has already seen *and also* any transmission with an **odd** sequence number.

True / False This scheme is secure against replay attacks. **True**
True / False This scheme is secure against reflection attacks. **True - having disjoint sequence number spaces means a message sent one way is not valid in the other direction**

(**B**) Alice and Bob return to their original protocol, but this time they want to stop keeping track of sequence numbers. Instead, they use a function `time()`, which returns a standard UNIX timestamp (the number of seconds that have elapsed since January 1, 1970). They replace their sequence numbers with a call to that function. I.e., Alice sends:

$$encrypt(k_1, m|\texttt{time()}) \mid MAC(k_1, m|\texttt{time()})$$

Bob does the same when he wants to send a message to Alice, but uses k_2 in place of k_1. I.e.,

$$encrypt(k_2, m|\texttt{time()}) \mid MAC(k_2, m|\texttt{time()})$$

Bob drops any packet from Alice that does not include the current time, and Alice drops any packet from Bob that does not include the current time. Assume that their clocks are synchronized and that there is zero latency between Alice and Bob.

True / False This scheme is secure against replay attacks. **False - Eve could replay a message within the same second**
True / False This scheme is secure against reflection attacks. **True**

Initials:

VIII DDoS Attacks and Botnets

19. [6 points]: Ben runs a webserver that allows users to issue HTTP requests that execute certain database queries. Some of these queries are computationally-intensive and take up to five minutes to complete. All other queries complete within 5 seconds.

Ben's server can handle up to 10,000 HTTP requests at once, provided that no more than 1,000 of them are for computationally-intensive queries.

Unfortunately, Eve is mounting a DDoS attack against him, using the HTTP flood attack discussed in lecture. Eve controls a botnet with 5,000 machines.

Which of the following will prevent Eve from overwhelming the computational resources on Ben's server?

(Circle True or False for each choice.)

(A) **True / False** Rate-limiting the incoming requests such that Ben's server only receives at most two requests per second. **True - two computationally-intensive requests per second = 600 max, since they complete within 5 minutes.**

(B) **True / False** Requiring that every HTTP request use a secure channel, e.g., run over HTTPS. **False - secure channels don't prevent DoSing**

(C) **True / False** Beefing up his web server so that the number of requests it can handle is 20,000, and the number of computationally-intensive queries it can handle is 5,000. **False - Eve's machines could launch multiple requests at once.**

20. [3 points]: Eve decides to attack Ben's network resources instead of his computational resources. Her botnet is capable of generating a total of 500 Mbytes/s of traffic. The network in front of Ben's webserver can handle 1 Gbyte/s of traffic. Assume that Eve does not add any machines to her botnet.

Is it possible for Eve's botnet to cause Ben's network to be overloaded? Give a **brief**, **one-line** reason why or why not.

YES. **NO.**

Reason: ____________________

Yes - To saturate Ben's network, Eve needs to amplify the amount of traffic she's able to produce at least two-fold. She can get this amplication with a DNS reflection attack or an Opt-ACK attack. Many students also noted that there could already be at least 500MB/s in the network already, in which case Eve doesn't need to do anything fancy. This is technically correct (and we gave credit for it), although it wasn't the scenario we were picturing.

Initials:

21. [4 points]: Consider the Torpig botnet as described in "Your Botnet is My Botnet: Analysis of a Botnet Takeover" by Stone-Gross, et al.

(Circle True or False for each choice.)

(**A**) **True / False** Law enforcement could compromise Torpig for one day by seizing the domain that the C&C server intends to use on a future day, thereby preventing the C&C server from registering it and thus preventing bots from communicating with C&C for that day.

False - The C&C server would switch to a new domain unless the law enforcement agency also masqueraded as the C&C server.

(**B**) **True / False** Once Torpig has compromised a user's machine, it may launch phishing attacks against that user. These attacks could be prevented by using techniques such as challenge-response protocols or sitekey, as discussed in lecture.

False - The threat model is different here than in lecture. Here, the client has been compromised.

End of Quiz 2

Please double check that you wrote your name on the front of the quiz, and circled your recitation section.

Initials: