Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.5840 Distributed System Engineering: Spring 2023

Exam I

Write your name on this cover sheet. If you tear out any sheets, please write your name on them. You have 80 minutes to complete this quiz.

Some questions 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 a question ambiguous, write down any assumptions you make. Write neatly. In order to receive full credit you must answer each question as precisely as possible.

You may use class notes, papers, and lab material. You may read them on your laptop, but you are not allowed to use any network. For example, you may not look at web sites, use ChatGPT, or communicate with anyone.

I (5)	II (10)	III (5)	IV (10)	V (5)	VI (10)	VII (12)	VIII (3)	Total (60)

Name:

Gradescope E-Mail Address:

I MapReduce

You have a small MapReduce computation whose Map phase produces the following six interme-
diate key/value data items:

```
a/1
c/2
b/3
c/4
b/5
b/6
```

R (the number of reduce tasks) is two, and the two reduce tasks are numbered zero and one. There
are two single-core worker machines available to run reduces. It takes a worker machine one
second to process each intermediate key/value pair, so if there were just one worker the Reduce
phase would last six seconds.

Section 3.1 of Dean and Ghemawat's MapReduce paper discusses hashing the keys of intermediate
data. The hash function takes an intermediate key as argument, and returns a number. Your job is
to pick the hash function that will cause the reduce phase to take the shortest amount of wall-clock
time for the above intermediate data. You do not have to worry about the behavior of the hash
function in other situations. Circle the best answer.

   1. **[5 points]:**

      A. a → 0, b → 0, c→0
      B. a → 0, b → 0, c→1
      C. a → 0, b → 1, c→0
      D. a → 0, b → 1, c→1
      E. All of the above choices result in the same amount of time.

II   GFS

Consider the paper *The Google File System* by Ghemawat *et al.*

    2. [5 points]: Explain briefly how GFS ensures that there is at most one active primary for a given chunk.

A GFS file starts out containing the single byte "a". The file's single chunk is replicated on three chunkservers. Client C1 opens the file and issues a write with offset zero and data "b" (i.e. C1 asks to replace the file's content with "b"), and waits for the reply from GFS. After seeing the reply to the first write, C1 issues a write with offset zero and data "c", and waits for the reply. At about the same time, client C2 runs three iterations of a loop that opens the file and reads it; C2 waits for the reply to each operation before starting the next. GFS replies with success to every one of C1's and C2's requests. There are no other client requests.

    3. [5 points]: What sequences of read results could C2 see? Circle either YES or NO for each sequence below.

      A. Sequence: a, a, a   YES   NO

      B. Sequence: a, b, a   YES   NO

      C. Sequence: a, c, a   YES   NO

      D. Sequence: c, b, a   YES   NO

      E. Sequence: c, b, b   YES   NO

III   VMware FT

The Design of a Practical System for Fault-Tolerant Virtual Machines, by Scales et al., says in Section 2.2 and Figure 2 that the Output Rule must be enforced for network packets sent to clients.

Consider the following scenario with a primary, backup, one client, and an output instruction i telling the primary's network card to send a TCP packet to the client. FT delays sending this packet until it receives an acknowledgment from the backup for i. The primary crashes either before or after receiving the backup's acknowledgment and thus either sends or does not send the delayed packet. When the backup takes over it will first replay its log, which includes an entry for i. In replay mode the backup will not generate output, including for instruction i. Then, after replaying the log, the backup becomes the new primary.

4. **[5 points]:** Even though FT doesn't send the TCP packet for i in replay mode, it is possible for a client to receive the TCP packet twice. Explain how.

IV   Raft

Refer to Ongaro and Ousterhout's *In Search of an Understandable Consensus Algorithm (Extended Version)*.

Consider the following optimization for Raft: instead of calling `persist()` whenever the log changes in the AppendEntries RPC handler, the implementation calls `persist()` in a background thread every 100ms to reduce the persistence overhead.

> **5. [5 points]:** Describe a sequence of events that would lead to the implementation committing two different entries at the same index.

Alyssa hears about a new cool technology: persistent memory (e.g., Intel Optane). Persistent memory behaves like a DRAM memory but is nonvolatile: it retains its content in the event of a crash—that is, if a server reboots, the content of persistent memory contains the values from before the crash. Alyssa equips each server in the Raft cluster with persistent memory.

> **6. [5 points]:** Alyssa stores all the variables listed in Figure 2 in a `Raft struct`. Alyssa modifies her Raft library to store this `Raft struct` in persistent memory at a well-known address. She removes the persistor and the calls to it. When a server reboots, it sets the `Raft struct` pointer to the well-known address, and initializes the volatile state to 0.
>
> Alyssa notices that this implementation can result in incorrect behavior. Explain why.

V   Raft and ZooKeeper

Refer to Ongaro and Ousterhout's *In Search of an Understandable Consensus Algorithm (Extended Version)* and *ZooKeeper: Wait-free coordination for Internet-scale systems* by Hunt, Konar, Junqueira, and Reed.

Ben wants to simplify his Raft implementation, so he decides to use ZooKeeper for leader election instead of implementing it in Raft. He removes the `RequestVote` RPC and the `votedFor` state from his implementation. In his new version of Raft, when a server wants to become leader in term `newTerm`, instead of becoming a Candidate and sending `RequestVoteRPCs`, the server attempts to `create()` a regular znode with path `/raftLeader/{newTerm}` in ZooKeeper. If the `create` fails (which happens when the znode already exists), the Raft server does not become leader. Otherwise, if the create succeeds, the Raft server becomes leader in `newTerm`. The rest of the Raft implementation (e.g. what happens after a server becomes a leader) is unchanged from Ben's initially correct version of Raft.

7. **[5 points]:** Explain how this modification to Raft can result in incorrect behavior.

VI   Linearizability

These questions concern the material from Lecture 9, Consistency and Linearizability.

A service's state consists of just one value, a string. There are two operations: append to the string, and read the entire string. In history diagrams,

```
|--Axyz--|
```

means a client issued a request to append "xyz" to the string, and

```
|--Rabxyz--|
```

means a client issued a read, and the response was "abxyz".

8. **[4 points]:** The service's string starts out empty. The only requests are those shown in the following history. Is this history linearizable?

```
|----Ax----|  |----Ab----|
        |---Ay---|
                    |---Ryxb---|
```

9. **[4 points]:** Again, the service's string starts out empty, and there are no other requests other than the ones shown in the following history. Which choices of XXX result in a linearizable history? For each possibility, circle either YES or NO.

```
   |------Axy------|
|----Aab----|  |----Aq----|
        |----RXXX----|
```

A.	xyab	YES	NO
B.	axbqy	YES	NO
C.	abq	YES	NO
D.	xyq	YES	NO

VII ZooKeeper

Refer to *ZooKeeper: Wait-free coordination for Internet-scale systems* by Hunt, Konar, Junqueira, and Reed.

Alyssa has a ZooKeeper installation with three ZooKeeper replica servers (a leader and two followers). Alyssa runs the following two programs, M3 and M4, each on a separate client machine, starting them at about the same time.

```
M3:
  create("/m3", "c3po")
  if exists("/m4") == NO:
    print "three"
  print "done"

M4:
  create("/m4", "r2d2")
  if exists("/m3") == NO:
    print "four"
  print "done"
```

The `create()` and `exists()` calls invoke the ZooKeeper client library. The znodes involved, `/m3` and `/m4`, start out not existing, and both `create()` calls succeed. The `create()` calls ask for normal znodes, not ephemeral. `exists()` has one of three results: it returns `YES` if the znode exists, `NO` if the znode does not exist, and it causes the program to immediately exit if there was an error (e.g., if ZooKeeper says it has terminated the session). The requests are all synchronous. There are no other client requests.

10. [3 points]: Can Alyssa see *both* M3 print "three" and M4 print "four"?

11. [3 points]: Is it possible for both programs to print just "done" alone?

Alyssa is experimenting with coordinator election using ZooKeeper. When she starts, no znodes exist on her ZooKeeper service (other than the root). Alyssa starts both these programs at about the same time on two different client machines:

Machine M1 executes the following code in single session:

```
if create("/coordinator", ephemeral=true):
  print "M1 won"
  create("/m1", "xyzzy")
  if exists("/m2") == YES:
    print "M1 sees M2"
  while true:
    act as coordinator...
else:
  exit
```

Machine M2 executes the following code in a single session:

```
if create("/coordinator", ephemeral=true):
  print "M2 won"
  create("/m2", "xyzzy")
  if exists("/m1") == YES:
    print "M2 sees M1"
  while true:
    act as coordinator...
else:
  exit
```

There is no client activity other than these two programs.

12. [3 points]: Could Alyssa see *both* "M1 won" and "M2 won"?

13. [3 points]: Could Alyssa see *both* "M1 sees M2" and "M2 sees M1"?

VIII   6.5840

14. **[1 points]:** Which lectures/papers should we definitely keep for future years?

   – MapReduce
   – RPC, Threads, and Go
   – GFS
   – VMware FT
   – Raft
   – Q+A on Lab 2A/2B
   – Linearizability testing
   – ZooKeeper
   – Chain Replication

15. **[1 points]:** Which lectures/papers should we omit?

   – MapReduce
   – RPC, Threads, and Go
   – GFS
   – VMware FT
   – Raft
   – Q+A on Lab 2A/2B
   – Linearizability testing
   – ZooKeeper
   – Chain Replication

16. **[1 points]:** What can we do to improve the course?

End of Exam I