Department of Electrical Engineering and Computer Science
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.1800 Computer Systems Engineering: Spring 2025

Quiz I

There are 31 questions and 12 pages in this quiz booklet. Answer each question according to the instructions given. You have two hours to answer the questions.

- The questions are organized loosely by topic. They are not ordered by difficulty nor by the number of points they are worth.
- **If you find a question ambiguous, write down any assumptions you make.** Be neat and legible.
- You are not required to explain your answers unless we have explicitly asked for an explanation. You may include an explanation with any answer for possible partial credit.
- Some students will be taking a make-up exam at a later date. **Do not** discuss this exam with anyone who has not already taken it.
- Write your name and kerberos ID in the space below. Write your initials at the bottom of each page.

This is an open-book, open-notes, open-laptop exam, but you may **NOT** use your laptop, or any other device, for communication with any other entity (person or machine).

Turn all network devices, including your phone, off.

Name: SOLUTIONS

Kerberos ID: SOLUTIONS **@mit.edu**

I. Naming

1. [8 points] In general, we can say that some name N *resolves to* some object X. Reversing that, we can say that an object X *is named by* some name N. In this question, we consider name/object relationships in DNS and Unix. Although there are many kinds of names in systems, we are focusing here on human-friendly names represented as legible character strings.

   Pick True or False for each of the following statements:

   a. [✓ **True**] / **False** DNS allows a single IP address to be named by multiple names.
   b. [✓ **True**] / **False** DNS allows a single name to resolve to multiple IP addresses.
   c. [✓ **True**] / **False** Unix allows a single file to be named by multiple names.
   d. **True** / [✓ **False**] Unix allows a single name to resolve to multiple files.
   e. [✓ **True**] / **False** A DNS client can ask a DNS server to resolve a given host name to an IP address.
   f. **True** / [✓ **False**] A DNS client can ask a DNS server to change which IP address is named by a given host name
   g. [✓ **True**] / **False** A Unix program can ask the file system to resolve a given file name to a file
   h. [✓ **True**] / **False** A Unix program can ask the file system to change which file is named by a given file name

(a) One machine can host multiple websites, each w/ unique domain names
(b) E.g., in a CDN
(c) Symbolic links
(d) This is not the abstraction provided by the Unix filesystem
(e) Standard DNS client/server setup
(f) A human in charge of a domain would do this
(g) This is what a filesystem is for!
(h) Another standard function of a filesystem

Initials: SOLUTIONS

II. Virtual memory

[10 points] Suppose program X has the following page table entries that map virtual to physical pages. Here, W is the read/write bit and P is the present bit.

virtual address	physical address	W	P
1	4	1	1
2	1	1	$\boxed{0}$
3	2	$\boxed{0}$	1

Pay attention to these two bits

All memory is initialized with 0 values. X executes the following three write calls (the syntax here is `Write(a, b)` where a is an address and b is a value):

```
Write(1, 10)
Write(2, 20)
Write(3, 30)
```

2. Which of the calls, if any, will cause an exception? **Circle ALL that apply**., or circle "None"

   a. Write(1, 10)

   [✓ b.] Write(2, 20)

   [✓ c.] Write(3, 30)

   d. None

P=0 causes an exception (a page fault) to pull this page into memory.

W=0 causes an exception b/c program cannot write to this address

3. After X terminates, what are the contents of physical addresses 1, 2, 3, 4, **in that order**?

20, 0, 0, 10

Result of write(2, 20). Note that this write succeeds despite causing an exception

Result of Write(3,30) (which doesn't succeed)

Nothing writes to physical address 3

Result of write(1, 10)

Initials: SOLUTIONS

III. Unix

[15 points] The Unix file system contains many components, but (of course) not all of them are used for every operation. Here is the scenario:

- A process creates and writes a new file named "foo" in the Unix file system.
- This new file will be created in the directory "/u/katrina/docs"
- Assume that the i-node for this directory (that will contain the new file) is already in memory.
- Assume that the directory itself contains only a few entries (~5).
- After the file is created, the process writes 17 bytes into it.

4. In this scenario, which of these kinds of components would be **read**? Select **all** the items that might be **read** somehow while creating and writing this specific file (but do not select any items that are not read).

   [✓ a.] I-list → If we read the i-node, we're reading part of the i-list
   [✓ b.] I-number → to index into i-list
   [✓ c.] I-node → Of parent directory
   [✓ d.] Disk block containing (only) i-nodes → i-list
   [✓ e.] Disk block containing (only) file data → parent directory
   File too small ← f. Disk block containing (only) indirect pointers
   [✓ g.] File descriptor → Write call doesn't "know" that the file of interest is the one it just created. Will read the FD given
   [✓ h.] File buffer → Reads are buffered

5. For the exact same scenario, which of these kinds of components would be **written**? Select **all** the items that might be **written** somehow (updated, changed) while creating and writing this specific file (but do not select any items that are not written).

   [✓ a.] I-list → Updating i-node
   b. I-number → No change required
   [✓ c.] I-node → Updated length, new pointers to data
   [✓ d.] Disk block containing (only) i-nodes → i-list
   [✓ e.] Disk block containing (only) file data → The file data
   f. Disk block containing (only) indirect pointers → Still too small
   [✓ g.] File descriptor → Allocated & updated on creation
   [✓ h.] File buffer → Writes buffered

Initials:

6. Pick True or False for each of the following statements:

   a. **[✓ True] / False** If you follow a shell command with &, the shell will make a new process in which to run the command while the shell goes on in parallel to read and execute the next command.

   Child process created to execute command; parent (shell) waits

   b. **True / [✓ False]** If you don't follow a shell command with &, the shell will not create a new process but rather run the command to completion in the shell process.

   Process can seek to beginning (byte 0) on the file descriptor. If it succeeds, it's an ordinary file

   c. **True / [✓ False]** A process can't tell whether its standard input is a pipe or an ordinary file.

   d. **[✓ True] / False** Checking the return value of the `fork()` function enables a child process to execute different instructions from its parent.

   e. **[✓ True] / False** Since a child process can write to all files that are open for its parent at the time of the fork, race conditions are possible where both parent and child are writing to the same file at roughly the same time.

IV. Concurrency and Bounded Buffers

[10 points] Ben has the job of building locks to avoid race conditions in his team's new operating system, TypOS. He figures that he will do this with a simple character variable, with "x" meaning the lock is available and "y" meaning the lock is held.

Ben's version of `release()` is:

```
release(lock):
     lock = "x"
```

and Ben's version of `acquire()` is:

```
acquire(lock):
     while lock != "x"
          do nothing
     lock = "y"
```

7. What is wrong with Ben's implementation? **Pick the BEST answer.**

   a. He should be using integers, not characters

   b. He should be testing the value of `lock` in `release()`

   c. He should be testing the value of `lock` in `release()`, and *not* testing the value of `lock` in `acquire()`

   [✓ d.] He needs an atomic swap or exchange to implement `acquire()`

   e. He should initialize locks with a third value that is neither "x" nor "y"

Initials: SOLUTIONS

Thanks to your assistance, Ben now has correctly-functioning implementations of `release()` and `acquire()`. He uses them to implement bounded buffers for interprocess communication in TypOS, with the following versions of `send` and `receive`:

```
send(bb, m):
 // each invocation of send has
  // its own local variable:
  // my_s_index (64-bit int)
  while True:
    acquire(bb.lock)
    if bb.in - bb.out < N:
      my_s_index = bb.in
      bb.in = bb.in + 1
      release(bb.lock)
⊛ ——  bb.buf[my_s_index mod N] = m
      return
    release(bb.lock)
```

```
receive(bb):
 // each invocation of receive has
  // its own local variables:
  // my_r_index (64-bit int)
  // m (message)
  while True:
    acquire(bb.lock)
     if bb.in > bb.out:
       my_r_index = bb.out
       bb.out = bb.out + 1
       release(bb.lock)

       acquire(bb.rec_lock)
       m = bb.buf[my_r_index mod N]
       release(bb.rec_lock)

       return m
     release(bb.lock)
```

8. Pick True or False for each of the following statements:

   a. **True / [✓ False]** The code is correct if there is one sender and one receiver executing at the same time.

   b. **True / [✓ False]** The code is correct if there is one sender and many receivers executing at the same time.

   c. **True / [✓ False]** The code is correct if there are many senders and one receiver executing at the same time.

   d. **True / [✓ False]** The code is correct if there are many senders and many receivers executing at the same time.

(a) With one sender & receiver, send can be interrupted at ⊛ and then try to read a message that hasn't been written

(b) – (d) are false because (a) is.

Initials: SOLUTIONS

V. Virtual machines

[6 points] Suppose we run two user-mode programs, A and B, which are independent (e.g., don't access any common files) on a Unix operating system, which is running in a virtual machine (VM).

Now a **single** bug (like a divide by zero, or a random memory write) happens in a **single** component of this system.

9. The single bug is in program A. Identify **ALL** the components potentially affected.
   - [✓ a.] Program A
   - b. Program B
   - c. Unix kernel
   - d. VM monitor

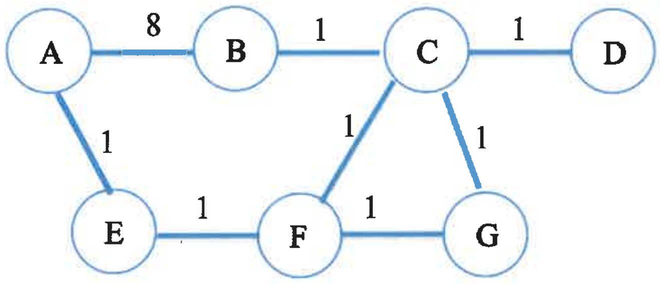
10. The single bug is in the Unix kernel. Identify **ALL** the components potentially affected.
    - [✓ a.] Program A
    - [✓ b.] Program B
    - [✓ c.] Unix kernel
    - d. VM monitor

11. The single bug is in the VM monitor. Identify **ALL** the components potentially affected.
    - [✓ a.] Program A
    - [✓ b.] Program B
    - [✓ c.] Unix kernel
    - [✓ d.] VM monitor

Initials: SOLUTIONS

VI. Routing

[14 points] Consider the network diagram below:



Alice implements a link-state routing algorithm on this network, using Dijkstra's algorithm and the shown link costs.

12. Using Alice's routing, how will a packet go from A to D? List the nodes in order, starting with A and ending with D. (Alternatively, you can write "Can't solve this problem on this network.")

A - E - F - C - D

13. Using Alice's routing, how will a packet go from G to A? List the nodes in order, starting with G and ending with A. (Alternatively, you can write "Can't solve this problem on this network.")

G - F - E - A

Bob now implements a link-state routing algorithm on this same network, also using Dijkstra's algorithm but using hop counts instead of the link costs in the diagram. (A hop count means that each link has a cost of 1 – every "hop" of the network increments the count)

14. Using Bob's routing, how will a packet go from A to D? List the nodes in order, starting with A and ending with D. (Alternatively, you can write "Can't solve this problem on this network.")

A - B - C - D

Initials: SOLUTIONS

In a (possibly unwise) compromise, the network operator has chosen to run Alice's routing implementation on the left-hand nodes (A, B, E, F) and Bob's routing implementation on the right-hand nodes (C, D, G). The two implementations send and receive identical-format advertisements containing all necessary information, but they continue to use different information to make their local routing decisions.

The net effect of all that is that Alice's nodes (A, B, E, F) make routing decisions identically to the ones you considered in questions 14 and 15, while Bob's nodes (C, D, G) make routing decisions identically to the ones you considered in question 16.

15. Using the combined routing, how will a packet go from A to D? List the nodes in order, starting with A and ending with D. (Alternatively, you can write "Can't solve this problem on this network.")

A – E – F – C – D

16. Using the combined routing, how will a packet go from D to A? List the nodes in order, starting with A and ending with D. (Alternatively, you can write "Can't solve this problem on this network.")

It won't – A routing loop will occur between B & C.

The combined network doesn't work very well, and the network operator is keen to get your advice on how to improve its routing behavior. They have identified a set of actions they are prepared to implement. They are not asking you to judge which ones might be better or worse choices, just to distinguish whether each one helps to make the routing work better, or not.

17. Which of these recommendations could improve the situation? **Circle ALL that apply.**

[✓ a.] Run Alice's routing algorithm on all the nodes

[✓ b.] Run Bob's routing algorithm on all the nodes

[✓ c.] Treat the Alice nodes and Bob nodes as two separate networks, and use BGP gateways to connect the differently-routed networks

[✓ d.] Ensure all links have identical costs

e. None of the above will improve the routing

These are all improvements in the sense that they eliminate routing loops like the one in #16.

Initials: SOLUTIONS

VII. Ethernet

18. [8 points] Pick True or False for each of the following statements:

  a. **True / [✓ False]** Ethernet receivers send acknowledgement (ACK) packets upon receipt of a packet that passes a CRC integrity check.
  b. **True / [✓ False]** Ethernet receivers discard duplicate packets.
  c. **[✓ True] / False** Ethernet transceivers implement distributed packet switching so as to avoid a single central point of failure.
  d. **True / [✓ False]** Ethernet's efficiency does not depend on how large the packets are.

VIII. RON vs CDN

[8 points] For each of the following phrases, choose whether it applies to RON, CDNs, both, or neither.

19. Intended to improve performance
RON / CDN / [✓ Both] / Neither

20. Value depends on presence in multiple AS's
RON / CDN / [✓ Both] / Neither

21. Dynamically updates model of likely latency along different potential routes
RON / CDN / [✓ Both] / Neither

22. Important for reducing tail latency of replicated workloads
RON / CDN / Both / [✓ Neither]

23. Distributes large content like images and movies to edge servers
RON / [✓ CDN] / Both / Neither

24. Packaged as a library for inclusion in distributed applications
[✓ RON] / CDN / Both / Neither

25. Rewrites URLs, redirects using DNS
RON / [✓ CDN] / Both / Neither

26. Risk of violating terms of service of the networks being used
[✓ RON] / CDN / Both / Neither

Initials:

IX. DCTCP

[12 points] We have a large-scale web application using the partition/aggregate design pattern with many workers and a requirement for real-time responsiveness. Assume there is NO background traffic (no other applications on the network) and all responses from workers are small (1-2 packets each).

27. Which of these problems is likely to be a concern in our network? **Circle ALL correct answers:**

    [✓ a.] Incast

    b. Queue buildup } Require small and large flows

    c. Buffer pressure

    d. None of these problems is likely

For the same application, Max has a bright idea of changing the aggregator structure. For example, instead of having one aggregator to aggregate the answer of 1000 workers, Max proposes using 10 aggregators each aggregating the answer from 100 workers, and then 1 aggregator to aggregate the answer from the 10 aggregators.

28. What is the most significant potential problem with this approach? **Circle the BEST answer**:

    a. It substantially increases the number of machines required

    [✓ b.] Doubling levels of aggregation might add unacceptable delay

    c. It creates longer queues

    d. It intensifies incast

29. TCP with RED/ECN has some similarities to DCTCP. What distinguishes DCTCP from TCP with RED/ECN? **Circle ALL correct statements that DISTINGUISH them:**

    a. DCTCP implements a central service to actively optimize the queue length of each switch. It does not

    b. DCTCP can be implemented in commercially-available switches. Both can

    c. DCTCP automatically determines the packet-marking threshold K. Neither does this automatically

    d. DCTCP does not adjust the size of the congestion window. Both do

    [✓ e.] None of the above statements correctly distinguishes DCTCP from TCP with RED/ECN.

Initials: SOLUTIONS

X. Physical deployability

[9 points] You are responsible for an application that runs on a collection of machines. The machines that currently run your application are in a single rack in a data center on the MIT campus. You use 30 machines in the single rack, each connected to the top-of-rack switch using Cat 6a cables, capable of 10 Gb/s over a distance of up to 100m.

You have been asked to evaluate a possible move to Fred's Discount Data Center, which offers lower prices. Fred is willing to offer you an empty rack to hold your 30 machines. However, part of his lower price is that he will connect your equipment to his top-of-rack switch using Cat 5 cables, capable of only 100 Mb/s over a distance of up to 100m. Note that 1 Gb = 1000 Mb.

30. Which of these issues will determine whether the move can work? A relevant issue is one where getting the "wrong" answer means the application likely won't work if moved into Fred's data center. **Circle ALL relevant issues.**

    [✓ a.] Does the application currently send/receive more than 100 Mb/s of network traffic at any node?

    [✓ b.] Is the currently-used MIT top-of-rack switch able to support more than 100 Mb/s per switch port?

    c. Do any of the machines use CPUs made by Intel? Irrelevant

    [✓ d.] What are the bandwidth and latency requirements of the application?

    e. None of the above are issues that will determine the success of the move.

Happily, Fred successfully addresses your concerns. Now it's almost time to move the servers! Fred admits that, unfortunately, he doesn't have an empty rack for you. Instead, he has three separate racks, each with space for 10 of your servers.

31. Which of these issues will determine whether the move can still work? A relevant issue is one where getting the "wrong" answer means the system likely won't work if moved into Fred's data center. **Circle ALL relevant issues.**

    [✓ a.] What is the available bandwidth between the racks?

    [✓ b.] What is the maximum network latency between the racks?

    c. Are the three racks different heights, or all the same height?

    [✓ d.] What are the bandwidth and latency requirements of the application?

    e. How long will it take to rewrite the application into three independent pieces?

    f. No additional questions needed, everything will be fine whether it's three racks or just one.

    g. The move may not work because of using three racks, but none of the above are issues that are relevant to the success of the move.

Initials: SOLUTIONS