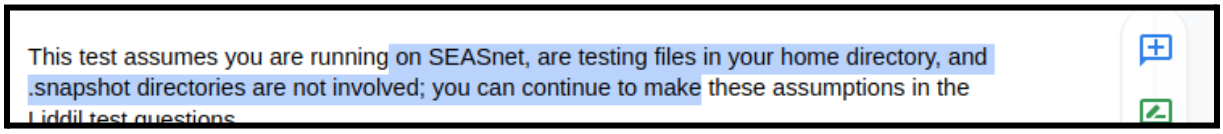
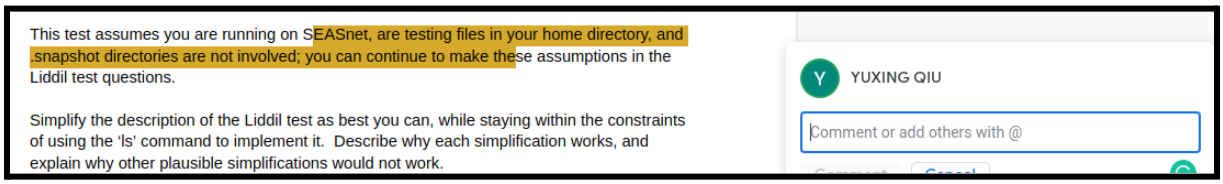
Final Review

(Hi all, if you would like to leave a comment for discussion, please **select the concrete sentence** before clicking the “comment button”. Otherwise, other people cannot figure out the context and you may not be able to get any meaningful replies. Thanks!

E.g: select this sentence, and click the comment button on the right side.



Then you may see sth like this:



1. Files, editing, and shells

File System

F20-1. The Liddil test checks **whether the strings A and B are known to name the same file**. It works this way:

* Run the shell command ‘ls -Liddil’ with the two operands A and B.

* If the command fails, the Liddil test fails.

* Otherwise, compare the two lines of output. The strings A and B name the same file if and only if the two lines are the same, except for the occurrences of ‘A’ and ‘B’ at the respective ends.

This test assumes you are running on SEASnet, are testing files in your home directory, and .snapshot directories are not involved; you can continue to make these assumptions in the Liddil test questions.

Simplify the description of the Liddil test as best you can, while staying within the constraints of using the ‘ls’ command to implement it. Describe why each simplification works, and explain why other plausible simplifications would not work.

Keys:

1. ‘ls’ command
    a. Basic command: list directory contents
    b. Options:
        i. -L: when showing file information for a symbolic link, show information for the file the link references rather than for the link itself
        ii. -i: print the index number of each file
        iii. -d: list directories themselves, not their contents
        iv. -l: use a long listing format
2. Goal: whether the strings A and B are known to name the same file, with the simplest method
    a. Why does the original command work?
        i. If
            1. Metadata (A) = Metadata (B)
            2. Inode (A) = Inode (B) (Memory Position)
        ii. Then
            1. A = B
    b. What is the most unique key of two different files
        i. Inode
        ii. Symlink (in this question, considered as referring to the same file)
    c. Multiple ways to refer to the same path

i. Relative path / Absolute path
ii. Redundant characters in paths:
    1. Multiple “/”s: //usr///bin/// ⇔ /usr/bin/
    2. “.” for current dir: /usr/./.bin/ ⇔ /usr/bin
    3. “..” for parent dir:
        a. /usr/bin/../home/ ⇔ /usr/home/
    4. “~” for home dir, “-” for the previous dir we just visited

---

F20-3. Suppose some other program is modifying the file system at the same time that your Liddil-test program runs.

How would this affect the validity of your program’s results? Briefly **explain**.

Goal: whether the strings A and B are known to name the same file, with the simplest method

- Why does the original command work?
- If
    - Metadata (A) = Metadata (B)
    - Inode (A) = Inode (B) (Memory Position)
- Then
    - A = B

How
Explanation

S20-1 (6pts)

1. Suppose you've written your own version of the cat command that, when called, outputs this to the terminal:

```
\    /\
 )  ( ')
(  /  )
 \(__)|
```

Assume that the full path name for the the directory where the executable for your personal cat command is stored is:

```
/u/cs/ugrad/bruin/better_commands/animals
```

1a (3 points). Suppose you're working on lnxsrv06 and decide this version of the command is good enough to be the default version of 'cat' for yourself. Write a one line command that will ensure that when you invoke the 'cat' command, you get back the personal version above.

Goal: default command, version-related problem

1. Symlink in /usr/bin;
    a. Achieve version controls
        i. eg we have multiple versions of gcc-x.x, but we can have a single gcc symlink in /usr/bin and have it point to a specific version gcc-m.n; Every time calling "gcc" will be equivalent to calling "gcc-m.n"
    b. If this command is originally an executable file installed in /usr/bin, then maybe rename the executable (in case we may need it later), and create a symlink after that
    c. May need "sudo"

2. Using Path
    a. What do you do when you have to use a different version of a given package or executable than what is in the path
    b. What is the use of export PATH=/usr/local/cs/bin:$PATH??
    c. How can you add the given path to use cat instead of the default path?
    d. Export PATH=/u/cs/ugrad/bruin/better_commands/animals:$PATH

```
1b (3 points).  Your friends aren't nearly as impressed by your cat
command as you thought they'd be, so you now decide to retire the
command.  You do not delete the executable; that is, the file stored
in the better_commands/animals folder still exists.  Write a command
that will ensure that when you now invoke the 'cat' command, *after*
the events of (a), you get back the actual Linux concatenate command.
```

The reverse of the previous question

Emacs

M-3 (10 minutes). Consider the standard Emacs Lisp function 'buffer-name', documented as follows:

(buffer-name &optional BUFFER)

Return the name of BUFFER, as a string. BUFFER defaults to the current buffer. Return nil if BUFFER has been killed.

Design and implement an Emacs Lisp command gps-buffer that acts like Homework 2's gps-line, except it displays and returns the string "Buffer B" where B is the current buffer's name, instead of displaying and returning a string like "Line 27/106".

Also, briefly explain why there's not much real-world utility for a gps-buffer command, even though gps-line arguably has some utility.

10 min, 2 parts: Write a Lisp Function + brief explain why

Difference between hard links and soft links:
https://www.youtube.com/watch?v=4-vye3QFTFo&ab_channel=AhmedAlkabary

2. Commands and Basic Scripting

Shell Scripting

F20-2
2. Write a shell command named 'Liddil-test' that quietly succeeds if given two operands for which the Liddil test succeeds, quietly fails

if given two operands for which the Liddil test fails, and fails with
a diagnostic otherwise. For example, on SEASnet, ‘Liddil-test /in/sbh
/usr/bin/bash’ should succeed, whereas ‘Liddil-test '' /’ should
quietly fail because the empty string does not name a file. When
given two operands your command should not generate any output: it
should merely succeed or fail, so that the user can write a shell
A command like this:

```
if Liddil-test /bin/sh /usr/bin/bash
then echo 'same file'
else echo 'differing files'
fi
```

without getting any stray messages. Your command should not take any
options, and should treat all operands as file names even if they
begin with ‘-’; for example, ‘Liddil-test -E .’ should quietly fail
unless you happen to have a symbolic link like ‘-E -> .’.

Shell scripting:

- How to get the operand values: built-in shell variables, $1 and $2 for the first and the second operand
- The wrapper of Linux command: ls -i
- Get the inode number in ${first}
- Compare the inode number with the if statement
- Return success / fail: exit

Regular expression

S20-2 (6pts)

```
2 (6 points).  You and your friend decide that you've had enough of
seeing USC on the news, and you'd like to create a spam filter to
filter out the unneeded acronym.  Write an extended regular expression
that matches any string that consists entirely of the uppercase
letters U and S and C, but with the following restrictions:

 * The string must have a length of six.

 * The string must have exactly two copies of each letter (two U, two
   S, two C).

 * The string must start with the letter U and end with the letter C.

 * The string must be the concatenation of two substrings, and in
   each substring every U must precede every S, and every S must
   precede every C.  (The two substrings can differ in length and one
   substring can be empty.)

Here are some examples of valid matches (one on each line):

  USCUSC
  USUSCC
  UUSSCC

Briefly explain why your regular expression is correct.

Hint: Try writing out the combinations as a tree that will help in
deriving the regular expression.
```

Make use of the hint:

1. try the combinations
2. as a tree

In detail:

basically, the answer should be a list of possibilities with the same form as "U(sth|sth|sth|...)C". The idea is to use a tree structure to list all possibilities.

(Here are all the possibilities I can think of:

just starting from U, and consider the next possibility: could be -U, or -S, or -C; if it's -U, then ...; if it's -S, then ..., if it's -C, then ...)

U-U-S-SCC

U-U-S-CSC

U-U-CSSC

U-S-USCC

U-S-S-UCC

U-S-S-CUC

U-S-C-USC

U-C-USSC

Then we can write either the "U(sth|sth|...)C" way.

Or, a more tree-structure thing like:

U(U.. | S.. | C..) C, where ".." represents the possibilities for the second letter being U, S, or C.

Then we fill up these ..

- the first "..":

  U(U **(S...|C...)** | S.. | C..)C

  then fill up those "..." :

  U(U (S ***(SC|CS)*** | C***SS***) | S.. | C..)C

- then fill up the second ".."

  // pass

- finally, fill up the last ".." to get the final answer

  // pass

S20-5a 5b 5c (6pts)

```
5. Zoom is working on a Python tool that extracts info from
password-protected meeting links that follow this format:

 * Links start with the base URL "https://zoom.us/j/" or
   "https://SUBDOMAIN.zoom.us/j/", where SUBDOMAIN is a non-empty
   string of lowercase ASCII letters.

 * The base URL is followed by a 9-digit conference ID (e.g.,
   356209714).

 * The conference ID is followed by "?pwd=", which is followed by a
   32-character alphanumeric ASCII password (e.g.,
   JZ9u3x6FVBumItd2PxOrbAApyCv860i3).

Here are some examples that match the format above:

https://zoom.us/j/356209714?pwd=JZ9u3x6FVBumItd2PxOrbAApyCv860i3
https://ucla.zoom.us/j/701495623?pwd=5915EqMgkRtaeCyrSTXouXeRnaQlSeCm

5a (2 points). Write a regular expression that matches with a valid
base URL.
```

Extended RE
https://([a-z]+\.)?zoom\.us/j/

```
5b (2 points). Write a regular expression that matches with a valid
9-digit conference ID.
```

[0-9]{9}

```
5c (2 points). Write a regular expression that matches with a valid
32-character alphanumeric password.
```

[0-9a-zA-Z]{32}

Advanced Commands

S20-3

```
3a (2 points). Write a Bash command that implements a configuration of
the Caesar cipher (a very simple substitution cipher). The command
should comply with the following specifications:

 * Input is from stdin.
 * Output is to stdout.
 * Only substitute lowercase and uppercase ASCII letters.
 * Use a right rotation of three places i.e. substitute ‘a’ with ‘d’,
   ‘b’ with ‘e’, ‘w’ with ‘z’, ‘x’ with ‘a’, and so on.  Similarly, do
   the same with uppercase letters: substitute ‘A’ with ‘D’, ‘B’ with
   ‘E’, ‘W’ with ‘Z’, ‘X’ with ‘A’, and so on.

For example:

   Input: afC2xA
  Output: diF2aD
```

What shell command is useful for substituting individual characters?
– > Likely sed or tr
If we want to use tr for simple character substitution, do we need to use any of its options?
– >likely no
Could you use grep on this question?
– > Possibly, but since we want to find + replace other tools may be more useful

tr '[a-zA-Z]' '[d-zabcD-ZABC]'

```
3b (2 points). Write another Bash command such that the previous
specifications apply, except:

 * Substitute lowercase letters ‘x’ through ‘z’ with the ‘@’ character
   instead of their previous substitutions.

 * Substitute uppercase letters ‘A’ through ‘W’ with the ‘@’ character
   instead of their previous substitutions.

For example:

   Input: afC2xA
  Output: di@2@@
```

tr '[a-zX-ZA-W]' '[d-z@@@ABC@@@@@@@@@@@@@@@@@@@@@@]'

there’s def a better way of doing this
sed command : https://www.youtube.com/watch?v=ixOiOS35HYg&ab_channel=UdemyTutorials
sed ‘s/[A-Wx-z]/@/g’ | tr ‘[X-Za-w]’ ‘[A-Cd-z]’

Lisp

Refer to section 1-Emacs for example question.

Reminder:

Print out HW2 gps-line.el as a reference for Lisp syntax:

- How to define and use variables
- How to define and call a function
- Print to stdout with “message” (format label)
- How to do general arithmetic operations
- If statement and loops
- …

---

3. Scripting and Construction

Python

S20-5d

```
5d (4 points). Write a Python function parse_zoom(link) that takes a
valid meeting link as a string argument, extracts the conference ID
and password using regex, and returns the values as a tuple.  The
outputs for the above examples would be:

('356209714', 'JZ9u3x6FVBumItd2PxOrbAApyCv860i3')
('701495623', '5915EqMgkRtaeCyrSTXouXeRnaQlSeCm')

Python's standard library provides the 're' module for regex
operations.  To search for a regular expression match in a string, use
re.search(pattern, string).  You can retrieve matched substrings in a
regular expression with the help of capture groups and the group()
function.  For example,

  import re
  s = "hello world"
  match = re.search("hello (.*)", s) # search for pattern in s
  match.group(1) # first capture group result, i.e., "world"
```

Make use of the given information, the re module
The only thing that's not that clear in the description is how the "group" works
group(1) matches the first parenthesized subgroup, in this case, it's (.*)

M-5 (10 minutes)
Write a Python function rotseq that takes two arguments, a sequence s and an integer n, and returns a list with the same elements as the sequence, except rotated by n items. For example:

rotseq(['a','b','c','d','e'], 0) should return ['a','b','c','d','e']
rotseq(('a','b','c','d','e'), 1) should return ['b','c','d','e','a']
rotseq('abcde', 4) should return ['e','a','b','c','d']
rotseq(['a','b','c','d','e'], 5) should return ['a','b','c','d','e']
rotseq('abcde', 6) should return ['b','c','d','e','a'].
rotseq(['a'], 10000) and rotseq('a', 10000) should both return ['a']
rotseq([], 10000) should return []

Negative n values should rotate in the opposite direction. For example, rotseq('abcde', -1) should return ['e','a','b','c','d'].
Implement your function as elegantly and simply as you can.

1. What is the pattern you see when you rotate?
2. What all possible inputs can you have?
3. What if the rotate value is greater than the size of the sequence?
4. Try running the above on different negative and positive numbers and see if you get a pattern.
5. What happens if you modify the array in place?

M-6 (10 minutes)

Write a Python program ‘dupsout’ that copies to standard output all input lines that are exact duplicates of earlier input lines. Here is an example of input (left column) and output (right column).

orangoutans	Argus
Emmeleen	Emmeleen
Argus	Emmeleen
Argus	orangoutans
Emmeleen	Emmeleen
Emmeleen	
chastenesses	
orangoutans	
Emmeleen	

1. If you have an array of words coming, how do you determine the next word is duplicate?
2. You will have to store the previous words somewhere.
3. Then check if the current word is present in the place, it is duplicate.
4. What are different data structures in python that we can utilize to make this implementation easy?

Interpreted/Compiled Language

---

4. Client-Server Apps and User Interfaces

Most questions focus on the understanding of

- The concept itself
- The pros and cons of the given concept
- The user cases of the given concept

Most questions ask for a general description.

What we can do before the final:

- Make our own summaries

User Interface

TCP/UDP

Example Question: From the point of view of developing your project, briefly explain the significant pros and cons of basing HTTP on TCP.

Features of TCP/UDP
Pros and cons of TCP/UDP
Google it and there will be lots of answers, print out those summaries, or your own summary, and bring it to the finals (if you cannot remember all details)

HTML & Web App

Example Question. In a client/server app, does out-of-order execution primarily benefit throughput or latency? Briefly explain.

- Features
- Pros and Cons

Google the concept mentioned during lectures, summarize the concept, and the pros and cons

In-order processors [edit]

In earlier processors, the processing of instructions is performed in an instruction cycle normally consisting of the following steps:

1. Instruction fetch.
2. If input operands are available (in processor registers, for instance), the instruction is dispatched to the appropriate functional unit. If one or more operands are unavailable during the current clock cycle (generally because they are being fetched from memory), the processor stalls until they are available.
3. The instruction is executed by the appropriate functional unit.
4. The functional unit writes the results back to the register file.

Often, an in-order processor will have a straightforward "bit vector" into which it is recorded which registers a pipeline that it will (eventually) write to. If any input operands have the corresponding bit set in this vector, the instruction stalls. Essentially, the vector performs a greatly simplified role of protecting against register hazards. Thus we observe that Out-of-order uses 2D Matrices where In-order uses a 1D vector for hazard avoidance.[*citation needed*]

Out-of-order processors [edit]

This new paradigm breaks up the processing of instructions into these steps:

1. Instruction fetch.
2. Instruction dispatch to an instruction queue (also called instruction buffer or reservation stations).
3. The instruction waits in the queue until its input operands are available. The instruction can leave the queue before older instructions.
4. The instruction is issued to the appropriate functional unit and executed by that unit.
5. The results are queued.
6. Only after all older instructions have their results written back to the register file, then this result is written back to the register file. This is called the graduation or retire stage.

Node.js and React

F20-8. Consider the following potential criticisms of Node.js:

* It doesn't support multithreaded applications, which makes many apps hard to scale.

* It's tied closely to Google's V8 JavaScript engine, and so is not portable to other platforms.

* It's too low level; if you want to build a real app you must write everything nearly from scratch, or pull in other peoples' code like crazy.

* JSX is too complicated and its learning curve is too steep, compared to other approaches.

Add one other reasonable criticism to this list.

For each of the five criticisms (the above four, plus your fifth):

* Argue that the criticism is a reasonable one, as best you can.

* Give a defense of Node.js against the criticism, as best you can.

* Explain how well the criticism applies to the project that you worked on for this course.

Node.js features, pros, and cons

F20-11. Suppose we want to reimplement Emacs using Node.js and React, instead of the existing Emacs code base that uses C as its core. We want **existing Emacs scripts and and keystrokes to work unchanged**, so that current users can continue to get their work done.

Does this idea make sense? Briefly explain why or why not. Either way, give the biggest obstacles you see to making it work.

Goal: existing Emacs scripts and keystrokes to work unchanged

- For emacs scripts
  - How to compile/interpret it
  - How to load it to Emacs
- Keystrokes
  - How to deal with user actions
  - How to deal with key bindings

---

5. Package Management

No such question in sample finals

Example question could be:

We have two different projects which would like to link to different python libraries, how to achieve that on a single machine?

- Use python virtual environment

6. Change Management, version control

Diff and Patches

S20-4

```
4. The following is a patch for a project you're working on.

  --- a/src/coolors.c  2020-05-31 09:14:41.000000000 -0700
  +++ b/src/coolors.c  2020-05-31 09:14:37.000000000 -0700
  @@ -1,12 +1,17 @@
  +/* Checks that an RGB parameter is valid */
  +int is_valid_rgb(int val) {
  +    return val >= 0 && val <= 255;
  +}
  +
   /* Convert RGB color to HEX format */
   int rgb2hex(int r, int g, int b) {
  -    if (r >= 0 & r <= 255) {
  +    if (is_valid_rgb(r)) {
           fprintf(stderr, "Invalid red value");
           return -1;
  -    } else if (g >= 0 & g <= 255) {
  +    } else if (is_valid_rgb(g)) {
           fprintf(stderr, "Invalid green value");
           return -1;
  -    } else if (b >= 0 & b <= 255) {
  +    } else if (is_valid_rgb(b)) {
           fprintf(stderr, "Invalid blue value");
           return -1;
       } else {

4a (1 point). Explain what this patch does at a high level.
```

diff of two versions of file src/coolors.c
Look at what lines are being removed and what lines are being added.

Try to understand what the change does
This is making the code modular.

```
4b (1 point). How many lines are added and/or removed by the patch?
```

Count the number of + and -

```
4c (2 points). Write a shell command that applies the patch. Assume
coolors.c is in src/coolors.c and the patch file, named cool.patch, is
in the root of the project directory.
```

What is a patch? The diff command examines two different versions of a file and lists the differences between them. The differences can be stored in a file called a patch file.
Use patch command to apply the patch to the given file.

patch src/coolors.c -i cool.patch
OR
git apply cool.patch

Git

F20-4. In .git/objects Git splits the 40-character SHA-1 text checksum into two parts, of length 2 and 38 characters respectively. Why doesn't Git instead split them into equal parts of 20 characters each? Wouldn't that be fairer or more efficient in some way? Or, of 2 vs 38 is good, why wouldn't 1 vs 39 (or 3 vs 37, etc.) be better? Briefly explain.

The file system has a limitation on the maximum number of files in a directory

- 1 vs 39 => more files in each folder

Searching/Scanning performance

- (Concrete explanation: https://stackoverflow.com/questions/18731887/why-does-git-store-objects-in-directories-with-the-first-two-characters-of-the-h)

F20-5. Data backup systems use deduplication, compression, and encryption heavily. Compare and contrast their use of these three technologies to how Git uses them. Assume Git plumbing only; do not worry about

porcelain or add-ons.

Deduplication: remove redundant files
Compression: remove duplicate data inside a file
Encryption: encode file with some unique keys

Git:
SHA hash from content: encryption
Bytecode to store object content: compression
Same content -> same hash ID -> same object: deduplication

Git Push/Pulls use SSH and, thus, are encrypted.

F20-9. The man page for 'git merge' has the following words of wisdom:

> Running git merge with non-trivial uncommitted changes is discouraged: while possible, it may leave you in a state that is hard to back out of in the case of a conflict.

Write a shell script that illustrates this advice. Your **shell script should set up a Git repository and working files from scratch (no fair cloning from any other source), and run 'git merge' with nontrivial uncommitted changes.** Explain the resulting state, and why this sort of state is hard to back out of in the more-typical case where you don't have a complete shell script that you can replay.

- Create a working folder and cd into it
- Git init
- Create working files, commit some changes
- Create new branches and edit some files (don't commit the changes)
- Git merge
    - with --abort we can ignore those uncommitted changes
    - But in this way, we cannot recover those uncommitted changes

A sad story:
https://stackoverflow.com/questions/26404604/is-there-a-way-to-recover-uncommitted-git-working-tree

---

F20-10. The man page for 'git rebase' does not have words of wisdom that are

like those mentioned in the previous (‘git merge’) question. Should it? Briefly explain.

Feature of rebasing
Difference between git merge and git rebase

It’s much harder to rebase with uncommitted changes (no force option like with git merge)
https://www.youtube.com/watch?v=kMvLn8WcAII&ab_channel=Ihatetomatoes

---

S20-10 (7pts)Cc

10. You are working on a software project that uses Git as the dedicated version control system. You clone the remote repository to a local directory on your computer. Your local repository has the following commit history:

C1 ← C2 (master, origin/master)

where C1 and C2 are commits and C2 is pointed to by the ‘master’ and ‘origin/master’ branch references.

The remote repository also has the following commit history:

C1 ← C2 (master)

The following questions depend on each other and should be answered sequentially.

10a (2 points). You commit a new snapshot to your local ‘master’ branch (assume the new commit is C3). What does your local commit history look like now? Make sure to draw the commit history (you can use the format above). Also, explicitly show the updated branch references and generated commits, if any.

C1 - C2 (origin/master) - C3 (master)

10b (2 points). Unfortunately, someone on your team has pushed new changes to the remote ‘master’ branch before you could push your local changes. The remote repository now looks like this:

C1 ← C2 ← C4 (master)

You pull from the remote repository. What does your local commit history look like now? Make sure to draw the commit history (you can use the format above). And explicitly show the updated branch references and generated commits, if any.

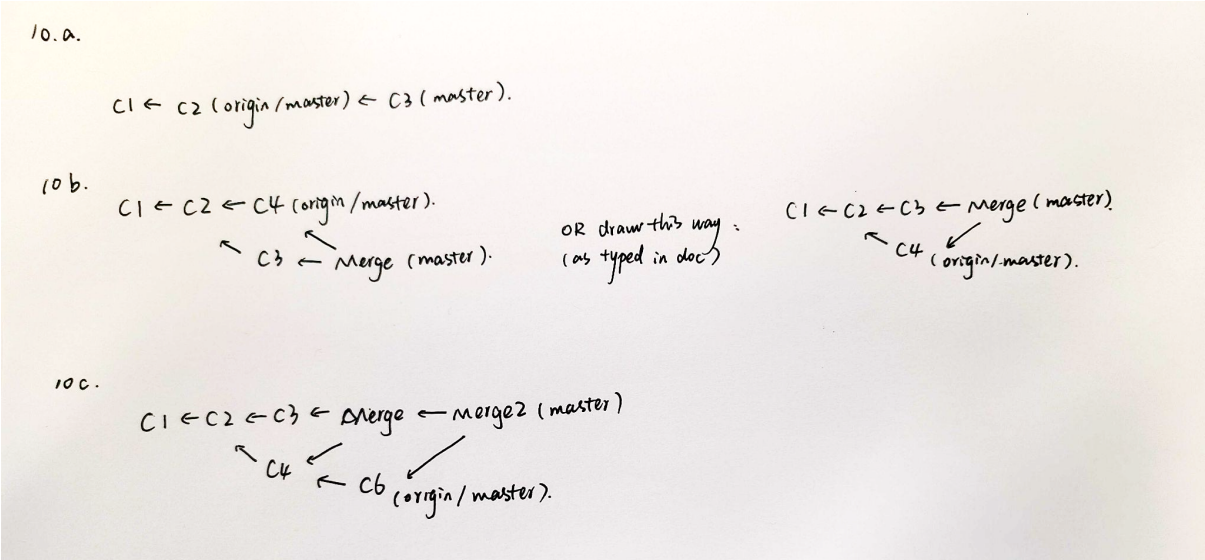
C1 - C2 - C3 ---------- Merge (master)
C2 - C4 (origin/master) -

10c (3 points). Once again, before you could push your local changes, someone on your team pushes their work to the remote repository. (Your team really needs to work on communication.) The remote repository now looks like this:

```
C1 ← C2 ← C4 ← C6 (master)
```

After pulling from the remote repository, what does your local commit history look like now? Make sure to draw the commit history (you can use the format above). And explicitly show the updated branch references and generated commits, if any.

C1 - C2 - C3 - Merge ----------Merge2(master)
C2 - C4-(connected to Merge and C6) - C6(origin/master) - Merge2



S20-11 (3pts)

11. Assume your local commit history looks like this:

```
    C2 (feature)
   /
  C1 ← C3 (master)
```

where C1, C2, C3 are commits and 'master', 'feature' are branch references.

11a (2 points). You are currently on the 'feature' branch. You run 'git rebase master'. What does your local commit history look like now? Make sure to draw the commit history (you can use the format above). And explicitly show the updated branch references and generated commits, if any.

C1-C3(m)-C2(f)

```
11b (1 point). You now checkout the master branch and invoke 'git
merge feature'. What type of merge occurs?
```

Fast-forward merge (for this question) v.s. Recursive merge

S20-13 (4pts)

```
[page 15]
13. For the following questions, use the commit graph below. Assume
that c1 is the oldest commit, and that each commit points to its child
commits.

c1 → c2 → c3 → c10
           \
            c4 → c7 → c8 → c9
             \        /
              c5 → c6

13a (2 points). Give a valid topological ordering of this graph. You
do  not need to list branch names, or use the sticky start/end
notation from Assignment 9.
```

HW6

Principles of topological order:

- If we have an edge (u => v), then u should come before v in the ordering
- Topological ordering is possible iff the graph has no directed cycles

```
13b (2 points).  Suppose I pull a copy of commit c7 to my own local
machine, and I use git log to read through commit messages for this
project. Of the 10 commits, which commit messages will I not be able
to read from my copy?
```

Things I can read：Everything on the current branch and comes earlier than C7

Not be able to read: NOT (the previous sentence)

---

7. Low-level Construction and Debugging

C

F20-6. Suppose you wanted to add checkpoint/restart capability to a working randall program running on SEASnet. The idea is that you run the program like this:

randall 100000000000 >output

and that if the system crashes while in the middle of a run, after it reboots you can continue where you left off by doing this:

randall 100000000000 >>output

Explain how to modify your randall implementation to support checkpoint/restart in this way by using functions like 'fseek', 'ftell', or 'lseek'. Use explicit code snippets in your explanation where appropriate. Explain what trouble your modifications would have when doing checkpoint/restart if the modified randall outputs to a pipe to some other program, instead of outputting to regular file.

Internet-based test: Google fseek, ftell, lseek
If prof. didn't introduce anything about these files, then skip this question.

Sth we can learn from this question:

- Review all the C functions or system calls that prof. mentioned, and those appear in our HWs (hw5)
- Prepare some simple examples for these functions, just in case you forget during finals

---

S20-6 (10pts)

6a (8 points). In the space available below, explain the differences between the two different 2D arrays declared below, arr1 and arr2. Focus on the underlying memory structure, any performance implications, and use-cases. Assume that this code appears at the start of a function body.

```
const int ROWS = 5;
const int COLS = 4;

//arr1
int **arr1 = malloc(ROWS * sizeof(int *));
for (int i = 0; i < ROWS; i ++)
    arr1[i] = malloc(COLS * sizeof(int));

//arr2
int arr2[ROWS][COLS];
```

There are some confusions: depending on whether the compiler will optimize the constant integers during compilation.

If all const ints are substituted during compiling time:

- Static array v.s. Dynamic memory allocation

- https://www.geeksforgeeks.org/difference-between-static-and-dynamic-memory-allocation-in-c/

Else

- Variable-length array v.s. Dynamic memory allocation
- https://stackoverflow.com/questions/31199566/whats-the-difference-between-a-vla-and-dynamic-memory-allocation-via-malloc

---

```
6b (2 points). Do the lines of code below make sense?  Why or why not?
Assume arr2 is the same as the one declared in part (a) above.

  int **arr3 = arr2;
  printf("%p", arr3[2][3]);
```

Nope, basically, they are of different types.
https://stackoverflow.com/questions/57756564/difference-in-function-parameter-double-pointer-vs-2d-array

---

S20-7 (4pts)

```
[page 9]
7 (4 points). On one running instance, the code below returns 10.
What are two possible reasons why?  Is either of those a concern and
why?

  return read(STDIN_FILENO, buffer, 50);
```

https://man7.org/linux/man-pages/man2/read.2.html

RETURN VALUE top

On success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number. It is not an error if this number is smaller than the number of bytes requested; this may happen for example because fewer bytes are actually available right now (maybe because we were close to end-of-file, or because we are reading from a pipe, or from a terminal), or because **read**() was interrupted by a signal. See also NOTES.

On error, -1 is returned, and *errno* is set to indicate the error. In this case, it is left unspecified whether the file position (if any) changes.

S20-9 (4pts)

```
You are a developer at a startup working on a project written in
C. Apart from the C standard library, the project also uses a
```

custom library you wrote called 'coolors'. Here's a code snippet from the project:

```
#include "coolors.h"
#include <stdlib.h>
 ...
/* Return a random hex color
*/ int random_hex_color() {
     int red = random_value(0, 256);
     int green = random_value(0,
     256); int blue = random
     value(0, 256);

                    _

     // Call function in coolors.
     int hex = rgb2hex(red, green,
     blue); return hex;
}
 ...
```

9a (2 points). You need to send the compiled executable to quality assurance (QA) for testing. Which linking method(s) would you use here to build the program and why? Assume that QA testers use the same architecture and Linux distribution that you do.

Difference between static and dynamic linking? If we need to send an executable how will both of them work? Really good illustration: https://medium.com/@bdov_/https-medium-com-bdov-c-dynamic-libraries-what-why-and-how-66cf777019a7

https://www.youtube.com/watch?v=YoyKDZlXCUM&ab_channel=embeddedarmdev

9b (2 points). Write a gcc command to compile the coolors source code, located in coolors.c, into a shared library, libcoolors.so

```
gcc -c coolors.c
gcc -shared -o libcoolors.so coolors.o
```

8. Open Questions

Basic idea:

- Describe the features/properties of all given concepts/method
- List the advantages and disadvantages of that concept/method, even use cases
- Don't make personal assumptions
    - If you have assumptions, make it clear
- Justify your choices

F20-12. Assuming you could pull off the idea of reimplementing Emacs in Node.js and React, how does it compare to the existing practice of using Emacs as a development environment for Node.js and React applications? Would there be significant advantages to the proposed implementation compared to the current one? Briefly explain.

S20-8 (6pts)

```
8 (6 points). One day you decide that it is annoying and confusing
that EOF is not a real character.  So you decide to make a new
language encoding, ASCII-with-EOF, where EOF is a real character.  You
do this by replacing ASCII character 28 (the file separator) with EOF
since the file separator control character is very rarely used in any
programming/files. You decide you won't miss it.

Now getchar() actually can return a char and read() will also fill in
EOF as a character in the buffer.

What are some cons of this idea? Would you want to permanently use
ASCII-with-EOF for all of your future work?
```

9. Security Basics

```
14.  Suppose you are worried that someone has broken into your SEASnet
GNU/Linux server and installed both a packet sniffer and a file
sniffer.  Both sniffers are malware in your server's operating system.
The packet sniffer snoops on all data sent to or from the network, and
the file sniffer snoops on all data being read from or written to
files.  While the sniffers are installed, you login into the server
via SSH and do Assignment 1.

14a (1 minute). Can the attacker now get a copy of your solution to
Assignment 1?  Briefly explain.
```

The file sniffer snoops on all data being read from or written to files

You login into the server and do assignment 1.

```
14b (2 minutes). Can the attacker now pretend to be you to log into
the same server via SSH?  Briefly explain.
```

The file sniffer snoops on all data sent to or from the network

It would depend on if you logged into the account before. Assuming you have, then, no, your login information would be encrypted and assuming it's not written to a file, there's no way for the sniffer to get this information.

```
14c (2 minutes). Can the attacker now set up a fake SSH server of
his own, one that you can mistakenly log into?  Briefly explain.
```

An attacker can do everything :)
But only with the given features, maybe not because the sniffer can only snoop data. The attacker may need to develop other tools to do sth like, stop your query and redirect it to the fake server.

Generally no. If the server was mimicking the same public key it would not be able to decrypt anything and authenticate its identity because it would not have the private key.