22S-COM SCI-35L-LEC-1 Midterm

[redacted]

TOTAL POINTS

85.5 / 100

QUESTION 1

1 1a **4.5 / 6**

- **0 pts** Correct
- **1 pts** Incorrect example.
- **2 pts** No attempt at an example

✓ **- 1.5 pts** **Close, but not correct explanation of lines that are matched.**

- **3 pts** Incorrect explanation of lines that are matched, but there was an attempt.
- **4 pts** No real attempt at an explanation.

Didn't mention that these parenthesis groups can occur multiple times on the line, separated by 0 or more spaces.

QUESTION 2

2 1b **4 / 4**

✓ **- 0 pts** **Correct**

- **2.5 pts** Incorrect simplification
- **1.5 pts** Incorrect or missing explanation.
- **4 pts** No attempt

QUESTION 3

3 2 **4 / 6**

- **0 pts** Correct

✓ **- 2 pts** **Incorrect explanation, but on the right track or explanation is too vague.**

- **4 pts** Incorrect explanation, or didn't explain _both_ why make doesn't allow circular dependencies and why Node does.
- **6 pts** No attempt

QUESTION 4

4 3a **10 / 10**

✓ **- 0 pts** **Correct or Almost Correct (very minor error)**

- **5 pts** Moderate Attempt (1-3 errors)
- **10 pts** Poor Attempt/Mssing

QUESTION 5

5 3b **10 / 10**

✓ **+ 7 pts** **Bracket Special Characters don't work (at least 2 examples or general concept)**

✓ **+ 3 pts** **Other characters will work (need to mention bracket special characters won't work or give specific examples of which characters will work or that the characters will be converted to ascii so they will generally work)**

- **+ 3 pts** Invalid ranges won't work
- **+ 3 pts** Other correct statements
- **+ 0 pts** Not correct/sufficient

QUESTION 6

6 4 **10 / 10**

- **3 pts** Partially Correct

✓ **- 0 pts** **Correct**

- **10 pts** Incorrect
- **2 pts** Partially correct
- **5 pts** partially correct
- **9 pts** Click here to replace this description.

QUESTION 7

7 5 **10 / 10**

✓ **- 0 pts** **Correct**

- **10 pts** Incorrect
- **3 pts** Partially incorrect
- **5 pts** Partially correct
- **8 pts** Click here to replace this description.

QUESTION 8

8 6 **8 / 8**

✓ **+ 4 pts** **Advantage: reliability (e.g., banking)**

✓ **+ 4 pts** **Disadvantage: head-of-line issue (e.g.,**

streaming)

+ **1 pts** Advantage - partial credit
+ **1 pts** Disadvantage - partial credit
+ **2 pts** Advantage - partial credit
+ **2 pts** Disadvantage - partial credit
+ **0 pts** Click here to replace this description.

QUESTION 9

9 7 **8 / 12**

✓ **+ 4 pts** **Bug 1: view-recenter should be after goto-char**

+ **4 pts** Bug 2: percent should be divided by 100

✓ **+ 4 pts** **Bug 3: the default value in the last line should be point-max**

+ **0 pts** Bugs were not found
+ **2 pts** Bug 1 partial credit
+ **2 pts** Bug 2 partial credit
+ **2 pts** Bug 3 partial credit
+ **2 pts** Key binding (I'm giving partial credit here since the code doesn't perform key binding as mentioned in the specification. However, performing key binding is not part of the function definition itself)

QUESTION 10

10 8a **6 / 6**

✓ **+ 4 pts** **Have several different modules in python application that rely on mymodule.**

✓ **+ 2 pts** **(Extension to rubrics 1) You import multiple of those modules in your code and thus 'import mymodule' is called multiple times**

+ **3 pts** Partially correct: Using in conditional statements
+ **2 pts** Partially correct explanation (example: Readability).
+ **1 pts** Incorrect but something relevant
+ **0 pts** No answer

QUESTION 11

11 8b **3 / 6**

✓ **+ 3 pts** **Module gets updated while running python code.**

Correct

+ **2 pts** Want to load the module (corrected version) again without restarting the whole code from scratch. (Example: While debugging)
+ **1 pts** Done when in interactive mode.
+ **2 pts** Partially Correct
+ **1 pts** Incorrect but something relevant
+ **0 pts** No answer

QUESTION 12

12 9 **4 / 6**

+ **3 pts** Correct: Briefly Explained: Renaming a symbolic link (Or moving it to another directory) points to a different file with the same name. (Use of mv command)

✓ **+ 2 pts** **Partially Correct: Briefly Explained: Renaming a symbolic link to point to a different file with the different name.**

+ **1.5 pts** Partially correct: Creating self loop. (The link becomes dangling)
+ **1 pts** Partially Correct: Explanation of symbolic link

✓ **+ 2 pts** **Correct: Briefly Explained: Why renaming hard links cannot change the file that it points to. (Points to inode)**

+ **1 pts** Partially Correct: Briefly Explained: Why renaming hard links cannot change the file that it points to.
+ **1 pts** (Correct) Possible with relative symlink and not absolute symlink
+ **0 pts** No answer

QUESTION 13

13 10 **4 / 6**

- **0 pts** Correct

✓ **- 2 pts** **Incorrect information about differences with debugging, but on the right track; a little to vague; or not enough detail.**

- **3 pts** Incorrect information about differences with debugging, or too vague.
- **4.5 pts** Didn't mention debugging differences.
- **6 pts** No attempt.

UCLA CS 35L midterm, spring 2022
100 minutes total, open book, open notes, closed computer.
1 minute = 1 point. Write answers on exam.

Name: [redacted] Student ID: [redacted]

1a (6 minutes). Explain what the following command does; give an example.

```
grep -E '^( *\([^)]+\))+ *$'
```

start esc esc end
any char other than)
anything but) spaces
spaces
$ ___(a+)+___
___(a+)

If matches a line starting with a bunch of spaces, followed by a (, any char other than), a +, then a closed). The line ends with another + and a bunch of spaces

1b (4 minutes). Simplify the following shell command, and briefly explain why your simplification works. It's OK if the simplification is slight.

```
sed 's/^ab/cdef/g'
```

sed 's/^ab/cdef/'

The 'g' is unnecessary because the '^' in grep will only match the start of a line and sed only reads in 1 line so there is only 1 possible matching location.

2 (6 minutes). Explain why Node can allow circular module dependencies (i.e., cycles in the module dependency graph), but a build tool like 'make' cannot allow circular dependencies.

make cannot because when it is packaging code up into an executable, circular loops will cause an endless loop of trying to inject code. However, JS is an interpreted language so node and the JS interpreter can figure out where to look without running into an endless loop.

3a (10 minutes). Write a simple shell command 'findrange' that accepts two arguments C and D, reads standard input, and outputs a copy of each line containing a character that is greater than or equal to C, and less than or equal to D. Your command can assume that C and D are both single printable ASCII characters. It should use case-sensitive comparison, using standard ASCII ordering; you need not worry about non-ASCII characters. Your implementation should use 'grep' to do the actual work of reading and writing lines.

For example, if standard input is ...	... then 'findrange w z' should output ...
The University of California requires that UCLA students, faculty and staff who are living, working or learning on campus or at other UCLA properties be vaccinated against COVID-19 - with limited exceptions.	The University of faculty and staff who are living, working or learning COVID-19 - with limited exceptions.
	... because the above are all the input lines that contain 'w', 'x', 'y', or 'z'.

findrange.sh

```
#!/bin/bash
grep "[$1-$2]"
```

3b (10 minutes). Explain what your 'findrange' does with arguments that specify special characters. For example, what does it do if its arguments are '"', '$', "'", '(', ')', '*', '+', '-', '.', '?', '[', '\', ']', '^', or '|'? If this exposes any bugs in your implementation, explain the bugs.

It just inserts them straight into the grep expression so they will be evaluated as part of it. Some characters will be unaffected such as (and) since they are part of a [] bracket group, but others will cause problems. For example, ^ will match the opposite which is not what we want and [or] will make the brackets unbalanced causing an error

4 (10 minutes). Suppose our version of Bash is buggy, because it consistently misspells 'export' with an 's' between the 'x' and the 'p'. That is, if you try to run the command 'export PATH' your attempt will fail with the diagnostic 'bash: export: command not found'. You can run the command 'exsport PATH' instead, and it successfully exports PATH.

You attempt to work around the problem by creating an executable shell script named 'export' in your PATH with the following two lines as its contents:

```
#!/bin/sh
exsport "$@"
```

Now when you run the command 'export PATH' in your buggy shell, you don't get a diagnostic and the 'export PATH' command exits with status zero, indicating success. However, the command still doesn't work like a standard 'export' command should. Explain what goes wrong and why.

While it works fine for a single argument, it won't work if you add arguments or try to use other features of the shell such as piping because it will pass all of that into exsport as the first param. Instead, you should do alias export=exsport to get the full functionality.

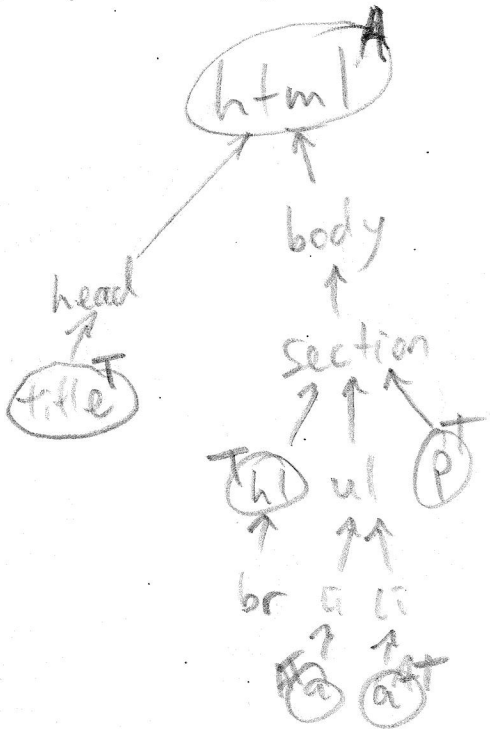
5 (10 minutes). Consider the following HTML taken from the class web page. Diagram the elements of the document tree that it represents. Draw arrows from each element to it parent element, if any. Label each element with its tag, along with the letter A if the element has attributes and the letter T if the element contains text that is not part of any sub-element.

```
<html lang='en'>
<head>
<title>UCLA Computer Science 35L, spring 2022</title>
</head>

<body>
<section>
<h1>UCLA Computer Science 35L, spring 2022.<br>Software
Construction</h1>

<ul>
<li><a href='news.html'>News</a></li>
<li><a href='syllabus.html'>Syllabus</a></li>
</ul>

<p>Lecture, 4 hours; laboratory, 2 hours; outside study, 6 hours.</p>
</section>
</body>
</html>
```



6 (8 minutes). HTTP running atop TCP/IP is good for some client-server applications, and not so good for others. Give an example of both the former and the latter and briefly justify your examples.

A good example is the CS35L website. It is the easiest way to make sure all the website data gets to us in order and saves the programmer the hassle of dealing with all that.

A bad example is video livestreaming because the TCP/IP will make the packets in order and make sure packets dont get dropped which could lead to freezing as it could wait a long time for a dropped packet from a while ago to arrive instead of dropping it and moving on.

7 (12 minutes). Fix three bugs in the following Emacs Lisp function, which is to be executed in View mode when you type '%'. The bugs cause the code to not agree with its documentation.

```
(defun View-goto-percent (&optional percent)
  "Move to end (or prefix PERCENT) of buffer.
Center display at point.
Also set the mark at the position where point was."
  (interactive "P")
  (push-mark)
  (view-recenter)
  (goto-char
   (if percent
       (+ (point-min)
          (floor
           (* (- (point-max) (point-min))
              (max 0 (min 100 (prefix-numeric-value
                               percent))))))
     (point-min))))
```

between 0 and 100

point-min + floor((pmax - pmin) * percent)

point-max

instead of point-min

should goto point-max if percent is not specified

Should push-mark and view-recenter after the goto-char instead of before

8. Normally in Python if you import the same module more than once, only the first import statement has an effect; the later import statements do nothing and take essentially zero time. If you want to actually import a module several times, you can use the reload function of the importlib module. For example:

```
import importlib
import mymodule
import mymodule # does no work
import mymodule # does no work
importlib.reload(mymodule)
importlib.reload(mymodule)
importlib.reload(mymodule)
```

8a (6 minutes). The above example is contrived; you'll never see those lines in practical code. Explain why you might want to use 'import mymodule' multiple times in a realistic Python application.

If you are writing a python application with multiple files, you might need the same module in multiple places, for example if my module was used in both a.py and b.py.

8b (6 minutes). Explain why you might want to use 'importlib.reload(mymodule)' instead of 'import mymodule' in a more realistic case.

If mymodule has some state that changes as you use it, you could run importlib.reload(mymodule) to reset the state back to its orginal value. Or if the module does some sort of calibration only once on import, you could reload to get it to recalibrate.

9 (6 minutes). Explain why renaming a symbolic link can change the file that it points to, whereas renaming a hard link cannot do that.

A hard link is a directory entry so when you rename it, you can only change the name of a file and not which file it points to.

However, a sym link is a file that points to another file in its contents, so when renaming it, you can change both the name of the symlink and its contents, which is the file it points to.

10 (6 minutes). Suppose React were written in C++ rather than in JavaScript. Explain how this would affect how you'd do Assignment 3 in React. In particular, how would you debug your program, compared to debugging your program with React as it is?

Everytime you made a change, you would have to recompile your project which would take much longer. To debug, you would need to use gdb instead of being able to look at values in the browser console.