COSC 3750
Linux Programming, Spring 2023
Homework 1, Using the shell and utilities.

The idea behind this assignment is that you should be able to figure out how to use some of the utilities we have talked about but more importantly, that you figure out how to read documentation for any utility and make it work. There will be situations where the documentation does not match the version of the utility or your misunderstanding and even rarer ones where what you want to do does not work because of a "feature" (bug).

There are six problems here. What I want you to turn in, **via WyoCourses**, is a single plain text file named exactly "username_hw1.txt". You WILL use either **tcsh** or **csh** to execute/test these lines. You do NOT have to change your shell. For instance, if you are using *bash* on your installation, just type *tcsh* at the prompt and now you are executing in *tcsh*. Type "exit" to get back to bash. The file will have one line for each problem. I should be able to copy each line and execute it in my shell to get the results specified. Remember, come up with a single command line for each problem. If you have any questions, please ask. The submission file can contain blank lines which separate the answers. Please number your answers and put them in order, smallest number to largest. Do NOT write some form of shell script. Do NOT search online for some else's version of how to do this especially StackOverflow. You figure it out.

Strangely, I can tell when you have not tested your work sufficiently, or at all. If you cannot get it to work, come by and talk to me. Emails to me will just not work for this because it is too easy for you to spend 2 minutes on it and then give up and send an email.

If you have questions on what the problem is asking, that I will answer those by email. Be warned, I do not usually read email after I leave the office, so late night before it is due means you are on your own.

Question 1: [10 points]

Use *find* to display the names of all files (starting in the current directory, any current directory) that have the ".tex" extension. This must recursively search the directories below the current one. Hint: you must use a wildcard for the filename and wildcards are problem you have to to solve. You need to actually test this by by creating several levels of directories and putting some ".tex" files in some of them. If you do not make a directory tree to test on, you are likely to get incorrect results.

Because you do NOT care what is in the files, you can create a ".tex" in some directory by simply doing

```
$> touch file1.tex
```

and voilá you have new empty file. And *find* does not care if it is empty or not.

Question 2: [10 points]

Use *grep* to find all of the lines in **some** files that contain the string "tomato". Your output will display the line numbers of the matching lines. For this one, use one or more filenames in place of the word **some**. Just read the manpage, this is simple. In this, **some** does not always mean use wildcards. I could just list four files on the command line. But I expect to see a line that shows that you tried it with more than one file at a time.

Question 3: [10 points]

Use *grep* and *wc* to count the number of lines in some files that do NOT contain the word "pepper". You should not use *grep*'s "-c" option because I want a single line of output from the command with a total line count regardless of the number of files processed at one time. Note: This will require a pipe. It will very definitely NOT require any temporary files or any utilities other that the two specified.

Some files is the same as the previous problem.

Question 4: [10 points]

Use *find* and *grep* to print the lines in the all the ".txt" files (like in the first problem) that have the string "Grade" in them. The output should include the name of the matching file(s) as well as the that actual matching line(s) of text. Do **not** number the lines. **READ** the man page on **find**. This will require the use of **find**'s "-exec" and "-name" options. This WILL NOT require any pipes or any other utilities besides *find* and *grep*. Be careful to protect all wildcards in filenames with single quotes. If you do NOT use **find** to locate the files, the answer is just wrong.

Question 5: [10 points]

Use *tr* to modify **some** text so that all uppercase characters are replaced with lowercase characters. You might consider that *tr* can read from standard input and most versions CANNOT read text directly from files. Maybe redirection? And again, **some** means just that, whatever I want to give it.

Question 6: [10 points]

Given **some** text, use *sort* to rearrange the text so that the lines are sorted numerically, from largest to smallest, using positions 2-6 (starting from the beginning of the line). The columns are NOT delimited by anything. I mean by that, positions on a line of the file where the character at the left margin is column 1, the next character (without any space) is column 2 and so on. This is not the same as having columns separated by spaces, tabs, commas, etc. Read carefully

Note that if you are sorting numerically, any line whose key does NOT contain an **initial** number is printed before all the lines that do have a numeric key. Those non-matching lines are printed in an arbitrary order, but it will probably be as they are encountered (or maybe sorted a-z).

In the normal numeric sort (no fields, no largest to smallest) matching lines are printed in numeric order with either the -n or -g switches. That means given

090
24sldkj

that 090 would be printed AFTER 24sldkj. This means that the **entire number** is considered and leading 0's are ignored.

Question 7: [10 points]

Given **some** text, display only the third column of data, sorted (alphabetically) in ascending order. Suppress duplicate values. The "columns" in this one are comma separated. You will use *cut* and then *sort*, and only *cut* and *sort* for this one. Again, use a pipe.