CSE 391, Summer 2020
Homework 7: More Regular Expressions and Sed
Due Tuesday, August 11, 2020, 11:00 PM

This assignment focuses on using regular expressions and related command `sed`. The set of files you will need will be on the latest master branch on the `cse391-20su-faang` repository in Gitlab. You aren't expected (and won't be able) to push any changes to the repository. You will only be able to pull down files that you need. You should assume you are in the `cse391-20su-faang` directory when doing these problems.

Submit your answers to Gradescope. In response to each question, **write the command that will perform the task described**, *not the output that the command produces*. Please be sure to write the **entire command** (including the command name and the input file). **Many of the necessary options to `sed` for this assignment are not supported on Macs so please be sure to use attu or the Linux VM.**

Task 1: Bash Shell Commands with `sed`:

After updating the website, adding cutting-edge products to its inventory and pivoting into the biotech space, FAANG has decided to rebrand to 24andMe, in attempt to establish itself as a leader in the field and beat an unnamed competitor. The company has also decided that this is a great opportunity to address tech debt. As part of the rebrand and code clean up, you will use `sed` to search and replace text in company files based on regular expressions.

Feel free to somewhat match your answers to the actual file you are given; your regexes do not have to work for a more general case. Write your commands in on the indicated lines in the `task1.sh` file. For many of the files, you should be updating them in place. We highly recommend providing a backup file extension to your sed command in these cases. If you want to discard your changes to a file and "reset" them back to the original state run the command `git checkout <file-you-want-to-reset>`. Be careful not to reset your task file!

1. You may have noticed that FAANG has yet to come up with a slogan. As part of the rebrand, the company wants to update its slogan. Write a command that replaces "Insert Catchy Slogan Here" in GenerateSite.java with whatever slogan you want, updating the `GenerateSite.java` file in place.
2. Selling dirt was poor for the company image. Write a command that replaces all occurrences of "dirt" case-insensitively, with "soil" in `Products.java`, updating the file in place.
3. The engineer who authored the `Product.java` and `Employee.java` gave the classes public fields. Their teammates are not impressed. Fix this style mistake by writing the command that replaces all instances of `public` with `private` *for the class fields only* in `Product.java` and `Employee.java`, updating the file in place.
4. Last week, the engineers discovered that the credit cards used on the site varied in formatting, causing a headache in validation. Some credit card numbers had spaces, some didn't and the spacing wasn't always consistent. Write the command that outputs the result of reformatting all of the credit cards from `cards.txt` to have consistent spacing, where each group of 4 numbers is separated by a single space and the last group of numbers may be 1-4 characters.
5. Java programs can contain single-line `//` comments and multi-line `/*` ... `*/` comments. Sometimes a programmer uses a multi-line comment syntax, but the comment only occupies a single line. Write a command that finds `/*` ... `*/` comments in `Products.java` **that occupy a single line** and replaces them with a `//` comment, in place. For example, `/* premium dirt */` would become `// premium dirt`. (Your command doesn't need to modify comments where the `/*` isn't on the same line as the `*/`. You may assume that any given line contains at most one comment.)
6. As part of the rebrand, all references to FAANG in the source code will need to be updated. Write a command that will find and replace all occurrences of FAANG, case-insensitively, found in all java files in the repository to 24AndMe. Your command should update all the files in place.
7. Human Resources lost the list of employees at the company and realized the profiles on the staff webpage were too cryptic to use for payroll. The IRS doesn't recognize "Kermit T Frog" as a valid wage-earning citizen. You'll be reconstructing the list of employees by looking at all of the engineers who have contributed to the repository. Write a

command that uses the output of `git log` to find all of the unique contributors, case-insensitively, to the repository. A "contributor" should be all of the text that appears after "`Author: `" in the git log. Authors with the same name but different emails should be considered distinct. The output should be written to a file called `contributors.txt`.

8. Using your `contributors.txt` file, output all of the unique contributors, case-insensitively, based on name (i.e. ignoring email).
9. Each of these contributors should also be given a company email. Use the usernames from the emails located between < > in `contributors.txt` to generate and output company emails of the form `username@24AndMe.com`. You can assume the username of an email is any character that comes before the @ symbol. Be sure to only produce each unique email once.

Hints: The regular expression syntax hints from the previous section on `grep` also apply to `sed`. Also recall that some special characters must be escaped by a `\` backslash to be used in a regex pattern. Remember that putting a 'g' at the end of your pattern, such as `s/oldpattern/newtext/g`, processes all matches on a line.