Lab 10: Regular Expressions (regex)

Introduction

In this lab, you'll get to use `grep` and `sed` to run some regular expressions you've written. Writing good regular expressions is like solving a puzzle!

You should make an `answers.txt` file to answer assignment questions. If a problem asks for a command, please include the complete command; I should be able to copy-paste your command to a terminal and run it.

WHAT IS HAPPENING

If you are stuck, try using https://regex101.com/ to figure out what's going on with your regular expression.

Recall that `grep` and `sed` by default use 'regular' regular expressions; you need to use a command-line argument to get them to understand PCRE-ish regexes.

Problem 0: Introduction to Regular Expressions

Head on over to https://regexone.com and complete all tutorial lessons. In a folder named Screenshots, submit images showcasing your solution for each of the final 10 lessons (Zero or more repetitions:Matches abc or def).

Problem 1: That filter problem, AGAIN

So it turns out that all that C++ could be done with `grep` instead!

1. Write a command to filter out lines beginning with '#'. (You can test it with `story-plain.txt`.)
2. Modify the previous command to filter out lines beginning with whitespace and then '#' as well. (You can test it with `story-space.txt`.)

Problem 2: Lk m, n vwls!

Write a `sed` command to remove the vowels from a file.

Problem 3: Counting Lines

Your friend happens to run a service that generates random numbers. (It's a very profitable startup.) Each user request is recorded in a log file with the following format:

```
YYYY-MM-DD HH:MM:SS number
```

Your friend wants to know how many requests in `numbers.log` were made in January on average across years. (Hint: `$ wc -l` will count the number of lines in files or STDIN; note: that's a bash command with a lowercase L option.)

Problem 4: Phone numbers

`phonebook.txt` contains phone numbers (surprise, surprise). But, those numbers are not written in a consistent format; I just put them in however I felt like at the time. It'd be nice to have those formatted in a consistent manner.

1. Write a `grep` command to match all the phone numbers in the file.

2. Write a `sed` command to format all the numbers like so: `(ddd) ddd-dddd`, in place, while leaving the rest untouched
   (Hint: backreferences are your friend!)
   (Hint 2: Sadly, `sed` doesn't have `\d`, so you'll have to use `[0-9]` instead.)

Note: Your regular expression for this problem should not exhaustively match the numbers (e.g., `(555) 123 - 4567|573-555-1234|314 342 6678)`. That's silly.

Epilogue

As with previous labs, your git repo on http://git-classes.mst.edu is your submission. Don't forget to commit and push all relevant files. Make sure you see everything you expect on GitLab!

We expect to see the following files on your master branch:

- `README.md`
- `answers.txt`
- `.gitignore`, if you chose to make one
- `story-plain.txt`
- `story-space.txt`
- `phonebook.txt`
- `numbers.log`