CSE 391, Winter 2020
Homework 6: Regular Expressions
Due Tuesday, February 25, 2020, 2:00PM

Special thanks to CSE 142 for the inspiration for task3

This assignment focuses on using regular expressions and related commands such as `grep`. A set of files you will need for this assignment are available in the file `hw6.zip`, found on the Homework page.

Submit your answers to Gradescope. For all tasks, **determine a single `bash` shell statement that will perform the operation(s) requested**, *not the output that the command produces*. Each solution must be a one-line shell statement, but you may use input/output redirection operators such as >, <, and `|`. Please be sure to write the **entire command** (including the command name and the input file) not just the regular expression required for the problem. Remember that the `man` pages are your friend!

Task 1: Warm-up Bash Shell Commands with `grep`:

The following are warm-up exercises to get some practice with regular expressions and `grep -E` or `egrep`. For these exercises you'll be searching from a file called `names.txt` that contains both single-word usernames as well as full names. You may assume the file contains one name per line. Write your commands on the indicated lines in the `task1.sh` file in the `hw6` folder.

1. Write the command that finds all usernames from `names.txt` that contain at least one numeric character.
2. Write the command that finds all usernames from `names.txt` that are exactly 4 characters long and consist only of alphabetic characters.
3. Write the command that finds all first and last names from `names.txt`. A name is an uppercase letter followed by one or more lowercase letters and first and last names are separated by a single space. Your command should not match against names that contain a middle name or initials.

Task 2: Bash Shell Commands with `grep`:

After the few weeks at FAANG, management has discovered that we need to start actually selling products to stay in business. You'll be helping develop regular expressions to help in customer account creation and billing. For each item below, **determine a single `bash` shell statement that will perform the operation(s) requested**. Write your commands on the indicated lines in the `task2.sh` file in the `hw6` folder.

4. **Email validation:** As part of account creation, customers must provide a valid email address. Write a command that prints out all valid email addresses from `emails.txt`. For the purposes of this assignment, a valid email address is defined as follows:
   - Between 1 and 16 lowercase letters, uppercase letters or digits
   - An @ symbol
   - A domain (e.g. gmail) that consists of any number of lowercase letters
   - A period
   - A tld (top level domain, e.g. com) that consists of 2 or more lowercase letters

   You may assume that `emails.txt` contains one email per line.
5. **Password validation:** Customers also must choose a strong password when creating an account. Write a command that prints out all valid passwords in the file `passwords.txt`. A valid password is defined as follows:
   - Must be at least 8 characters long
   - Must contain at least one uppercase character
   - Must contain at least one lowercase character

- Must contain at least one digit

You may assume that `passwords.txt` contains one password per line.

6. **Credit Card validation:** In order to process payments for orders, FAANG needs to validate credit cards entered on the site. Since we are a small company, we will only be accepting Visa and Mastercard (Sorry American Express, your fees are too high.) Write a command that identifies all well-formatted credit cards in a file called `cards.txt` that FAANG accepts. The format of these cards are as follows:
   Master Card
   - Begins with a 5
   - Exactly 16 digits long, grouped into sets of 4 separated by a space

   Visa
   - Begins with a 4
   - Between 13 and 16 digits long, grouped into sets of 4 (last may be shorter) separated by a space

   You may assume that `cards.txt` contains one credit card per line. *Note:* true credit card validation also requires using a formula to validate the contents of the credit card number, in addition to basic formatting. You do not need to do this.

Task 3: Bash Shell Commands with `grep` to process genetic data:

The board is concerned about FAANG's product-market fit. As it turns out, dirt and peas (especially those not in a pod) don't have high market value. Therefore, in addition to continuing to sell its pre-existing products, FAANG has decided to enter the biotech space and process genomic data.

For these problems, you will be processing DNA data from the file `dna.txt`. Data is printed in the file in pairs of lines. The first line in the pair is the name of the DNA sequence and the second line is the DNA sequence itself. The following provides you with some context for the task, but an understanding of DNA is not required for this assignment.

DNA consists of long chains of chemical compounds called nucleotides. Four nucleotides are present in DNA: Adenine (**A**), Cytosine (**C**), Guanine (**G**), and Thymine (**T**). Certain regions of the DNA are called genes. Most genes encode instructions for building proteins (they're called "protein-coding" genes). These proteins are responsible for carrying out most of the life processes of the organism. Nucleotides in a gene are organized into codons. Codons are groups of three nucleotides and are written as the first letters of their nucleotides (e.g., TAC or GGA). Each codon uniquely encodes a single amino acid, a building block of proteins.

For these problems you will be identifying protein-encoding genes as well as other attributes of the genetic data in the file. **Note that all matches for these problems will be case-insensitive.** Write your commands on the indicated lines in the `task3.sh` file in the `hw6` folder.

7. Print all of the DNA sequences in the file (all of the non-names).

8. Print all of the names of DNA sequences in the file. *Hint: This will require use of a grep option, in addition to regular expressions.*

9. Print the full DNA sequences that contain the word "CAT", preceded by the name of the sequence. *Hint: man grep to look for options that alter what gets printed for a match.*

10. Print all of the DNA sequences that start and end with the same codon. (For this case, do not worry about matching start and end codons that don't have the same casing as each other.)

11. Print all of the valid genes in the file. That is, all of the DNA sequences that can be divided into codons. (i.e. A sequence of A, C, G and Ts and has a length that is a multiple of 3).

12. Protein-coding genes are genes that begin with ATG, end with one of TAA, TAG, or TGA, and are at least 5 codons long. Print all protein-encoding genes in the file.

Hints: Recall the **regular expression syntax** shown in class, such as a `.` for any character, `|` for "or," `[]` for character classes, `{}`, `*`, and + for quantities, `^` and `$` for specifying the start/end of a line, and `\<` and `\>` for specifying the start/end of a word. Also note that `grep` can use "back-references" to refer to sub-patterns captured previously between `(` and `)`, such as `\1` for the first captured sub-pattern, `\2` for the second, etc.