CS 279 - Homework 6

Deadline:

Due by 11:59 pm on **SUNDAY**, October 5.

How to submit:

Submit your files using `~ah270/279submit` on nrs-labs, with a homework number of `6`, by the deadline shown above.

Purpose

To practice with regular expressions and with bash shell scripts using command-line arguments

Important notes:

- **Each** bash shell script that you write is expected to include a descriptive opening comment block including your name and the last modified date.
- It is possible that your answers may be collected and posted to the course Moodle site.

The Problems:

Problem 1:

In a file `hw6-1.txt`, include:

- your name
- the part you are giving an answer for
- a regular expression for each of the following
  - Note: It would be a very good idea to test your answers using `grep` or `egrep` (whichever is more appropriate). Remember to use quotes as needed around your regular expression when needed during such testing.

1 part a

Write a BRE pattern that will match any line starting with `Romeo:` (this might grab each of Romeo's first lines from a script, for example)

1 part b

Consider the character class `[[:blank:]]` -- it matches (course text, p. 76) "the characters that produce horizontal whitespace" -- this might vary a bit based on the locale, but for POSIX these are

space and tab.

Appropriately using this character class (written as a bracket expression), write a BRE pattern that will match any line **ending** with 1 or more instances of horizontal whitespace.

1 part c

Using an appropriate subexpression, write a BRE pattern that will match any line containing a repeated integer.

1 part d

Using an appropriate interval expression, write a BRE pattern that will match `CS` followed by a blank followed by a 3-digit number.

1 part e

Using an appropriate interval expression and an appropriate subexpression, write a BRE pattern that will match any line that contains some repeated string of lowercase letters of length 3 or longer.

1 part f

Write a BRE pattern that will match any line ending with a semicolon.

1 part g

Write an ERE pattern that will match any line ending with a semicolon OR a curly brace OR a closing parenthesis. (Test this one using `egrep` or `grep -E`)

1 part h

Write an ERE pattern that will match any line ending with an integer (where that integer MAY, but is not required, to start with a + or -). (Test this one using `egrep` or `grep -E`)

HINT: backquote that + you are trying to match -- + is a special character in ERE!

1 part i

Write an ERE pattern that will match any line that starts with an integer (where the integer MAY, but is not required, to start with a + or -).

1 part j

And now write an ERE pattern that will match any line that starts with an integer (where the integer MAY, but it not required, to start with a + or -) followed by at least one non-digit character.

Submit your resulting `hw6-1.txt`.

Problem 2:

(adapted from University of Washington CSE 390, Spring 2010 Assignment 5: Basic Shell Scripting)

This is the first version of a script that we will refactor/improve later on.

Write a bash schell script `mantra1.sh` that accepts two command-line arguments:

- a string for a message to print
- a number of times to print it

The script should output/display:

- an opening "border" line of some style you choose
- then that message that many times,
- then a closing "border" line of some style you choose

For example,

```
mantra1.sh 'Moo moo moo!' 5
```

should result in something like:

```
========================================
Moo moo moo!
Moo moo moo!
Moo moo moo!
Moo moo moo!
Moo moo moo!
========================================
```

For this first version, your shell script can be very trusting -- you may ASSUME that:

- the user WILL pass 2 command-line arguments
- their values WILL be reasonable --
  - the message WILL be non-empty, and shorter in length that the width of your terminal
  - (the user WILL surround it with single quotes if it contains special characters...)
  - the number of time WILL be a valid integer greater than 0

Submit your resulting `mantra1.sh`.

Problem 3:

Consider your bash shell script `gen-fodder.sh` from Homework 5, Problem 3. Refactor it into `gen-fodder2.sh` so that it now accepts two command-line arguments -- which you can assume are:

- an integer greater than 0
- a desired final file line

...and it then creates the specified number of `test*.txt` files, rather than always creating 20, and in addition to the two lines it already puts in these files, it also puts as a third and final line the specified

second command-line argument.

Then, demonstrate its use as follows:

- create an empty new directory
- in that directory, run:
  - echo an "about to start test" message into a file `gen-test2.txt` (if you would like to append a blank line or some kind of border after this, you may do so)
  - append the results of an `ls` command to `gen-test2.txt` to show the directory's initial contents (just `gen-test2.txt`, at this point, and *maybe* `gen-fodder2.sh`, although you could also call it from another directory)
- now run `gen-fodder2.sh` in that directory, with appropriate command-line arguments of your choice
- and afterwards, echo an "after test" message and append it to `gen-test2.txt` (if you would like to append a blank line or some kind of border after this, you may do so)
- and then append the results of another `ls` command to `gen-test2.txt` to show the directory's resulting contents (should be considerably more files now)

Submit your resulting `gen-fodder2.sh` and `gen-test2.txt` .

Problem 4:

Recall the UNIX command `wc` - in the words of its `man` page, it displays to standard output "the number of lines, words, and bytes contained in each input file, or standard input (if no file is specified)".

Also consider -- which is your favorite UNIX text editor so far?

With that in mind, write a shell script `edit-all.sh` that meets the following specifications:

- for each command-line argument given, it should:
  - print a message noting the name of the file it is about to try to edit/create
  - then call your favorite UNIX text editor with the name of that file
- After doing that for all of the command-line arguments, it should:
  - print a message noting that it is about to show some statistics for these files, and then
  - call `wc` for each of the command-line arguments, thus showing their post-editing number of lines, words, and bytes.
- and close with a final farewell message of your choice

Submit your resulting `edit-all.sh`.