CS2043 Assignment 3

Due: Friday February 20th 2015 at 11:59 PM on `http://cms.csuglab.cornell.edu`.

You can form groups of two students to complete this assignment. Please form a group on CMS and submit only one solution per group.

General Note: This assignment (and future ones) require you to have access to a Unix-like (Linux, Mac OS X, etc) machine. If you do not have such an operating system installed on your local machine, make sure to get a CSUG Lab account. Different systems have slightly different configurations. The main environment in this class is GNU/Linux.

Assignment Notes:

- You must complete the assignment using GNU/Linux tools that were discussed in class.
- This assignment is composed of 6 parts.
- For the first 3 parts, treat upper and lower case characters the same and ignore punctuation marks.
- **Note:** Words are not just lists of alphabet letters. You should also match words that contain an apostrophe like: *don't, Frank's, can't* ...etc. But not arbitrary instances of single quotes as in *'this'*.
- For each problem, you will write a script (not the output thereof), and save it to a file named with the specified problem label, e.g., **problem1name.sh**. Assume that the input is in the same directory as the script. Remember that a bash script is a text file with: `#! /bin/bash`
  as the first line.
- Start this assignment early. It is longer and more challenging than the previous ones!

Text play: Frankenstein again

- The plain text version of Frankenstein is available at
  `http://www.cs.cornell.edu/courses/cs2043/2015sp/assignments/frankenstein.txt`
- A list of English prepositions was taken from Wikipedia and made available at
  `http://www.cs.cornell.edu/courses/cs2043/2015sp/assignments/prepositions.txt`

- **frankenprepos.sh:** Get the 100 most common words in the text of Frankenstein that are not prepositions (use the given list: `prepositions.txt`).
  Hint: You can build upon the `frankenread.sh` script you wrote for the previous assignment.

HTML scraping

– An HTML page of a recent New York Times article about the Super Bowl is available at: `http://www.cs.cornell.edu/courses/cs2043/2015sp/assignments/superbowl.html`

– For the purposes of this assignment, HTML text is anything that is not a tag. Meaning, `<someTag>`*text to scrape*`</someTag>`. You do not need to worry about JavaScript or other things that you might catch. We will consider all of that to be text for the purposes of this assignment. One easy way to identify tags here is that they begin with a `<` and contain characters that are **not** `>`.

- **webcrawler.sh:** Write a script to download and parse out the text (from HTML) of the provided NY Times article
- **commonwords.sh:** Get the 100 most used words in the text (you can use your prior scripts)
- **teamwords.sh:** Write a script that takes parameters `nbefore` and `nafter` which prints out the requested number of words that appear right before and right after any mention of the following words in the text: "seahawks", "patriots".

Spell Checker

In this exercise you will build a rudimentary spell checker for formal English using `grep`. Here is our strategy. We are going to tell `grep` to read the list of English words from the file at:
`http://www.cs.cornell.edu/Courses/cs2043/2015sp/assignments/english.txt.gz`.
We will tell `grep` to ignore case, and to invert the match so that it will only print lines that do not match the English words (that is, we delete the words that are correctly spelled and print the misspelled ones). We will also tell `grep` to match full words (and not substrings, such that futon does not count as a match for the preposition on). Consult the man pages for `grep` and find out how to accomplish these tasks.

- **spellchecher.sh:** Write a script that takes in an input text in formal English (i.e., without contractions, such as dont, were, theyre, and so on) and prints a list of the misspelled words. Warning: Your spell checker will not be perfect as there are many

missing words in the English words file. We will be looking for a correct script for this task, not for a perfect spell checker.

Data processing

- This part is mainly to help give you a taste of `awk`/`gawk`.
- A plain text, comma separated value, file containing an activities log for some person *(not me .. I am way too lazy for that)* is available at:
  `http://www.cs.cornell.edu/courses/cs2043/2015sp/assignments/activity_log.csv`
- The log has 3 columns: date, activity title, and time. Values from different columns are separated by commas.
- There are three types of activities: *work*, *run*, and *farmers market*.
- Each activity entry marks either the starting time of an activity or the end time. The notation used is: `start <activity>` and `end <activity>`.

- **activitylog.sh:** Calculate the total number of hours spent on each activity during the period in the log.

Remix

A remix is a song that has been edited or completely recreated to sound different from the original version. One common form of remix is to combine two songs together.

- **songremix.sh:** Assume that the input consists of two files, one for each song, whose names are `song1.txt` and `song2.txt`. Write a script to create remix lyrics that alternate between the lyrics of two different songs, i.e., first line of the remix is the first line of `song1.txt`, the second, the first line of `song2.txt`, the third, the second line of `song1.txt`, and so on. (If one song is shorter than the other, the longer should continue through its end, alternating with empty lines). We have made available a tarball with the lyrics of the top 100 songs on the radio today so that you can have fun with your script. Once you find two songs you want to remix, pass the two input filenames as arguments to your script and see what comes out. The lyrics tarball can be found at:
  `http://www.cs.cornell.edu/Courses/cs2043/2015sp/assignments/top100songsFeb2015.tar.gz`

  Remark: The lyrics files were automatically extracted from the web and may contain errors, such as missing new lines. Feel free to modify them so that they will look better after your remix.

Phonebook

We asked our friends to fill out a form with two fields, their names and their phone numbers. We collected 25 responses, but each person entered their numbers using a different format. For example, some of them wrote their number without area code, such as 9232-0092, others wrote them with separators, such as (607) 434-2323, and others without them, such as 905993232. A phone number contains between 7 and 11 digits.

- **phonebook.sh:** Download a small address book from: `http://www.cs.cornell.edu/courses/cs2043/2015sp/assignments/phone-data.txt.gz` and use `grep` to detect all the variations in phone number format.

Remember your two best friends are the `man` tool and your favorite search engine.
If you need to ask questions, use the class discussion board on Piazza.