Week 7 Assignment

Objectives:

Design and use regular expressions

Description:

In this assignment you will form various regular expressions. We will be working with the wordlist found on most linux installations. Mine is found at `/usr/share/dict/words`. If you do not have this file you can install it with a `sudo apt-get install wamerican`. All of the regular expressions can be tested by doing the following command: `egrep "EXPRESSION" /usr/share/dict/words`. There may be more than one answer for a particular question. You should NOT use any other pipes. All results should be contained within a single regular expression unless otherwise indicated. Each question will be coded into a bash function. I.e. the first regular expression would be put in a function named `one`.

1. Here is problem one that I am going to walk you through. Search for all words that begin with the string “foo” and that end with an “s”, I would type the command:

egrep “^foo.*s$” /usr/share/dict/words

I would look at it to see if I am getting the desired results. Once I am sure that I am getting the desired results I would put that in my function like this:

```
    function one () {
      egrep "^foo.*s$" /usr/share/dict/words
    }
```

You may create a `main` function to call all your other ones, but you should call main as explained in previous assignments.

Continue for all the questions below:

1. Search for all words that begin with the string “car” and that end with an “s” (as shown above)
2. Look for all words that contain the string “rya”.
3. Find all words that contain the string “mail” or “chick”.
4. Find all words that have 8 successive characters that are consonants
5. Find all words that have exactly 15 characters.
6. Find all words that end with ‘ux’.
7. Find all words that start with a ‘b’ or a ‘s’ and have a ‘zz’ somewhere later on in the word.
8. Try this expression: `^(.).\1$` . The ^ matches the beginning of the line. The . means any character. The . within ( ) means capture this as a sub-pattern. The next . matches any character. The \1 refers to the pattern captured in the ( ). The $ means match the end of the line. Now find all five letter words that start and end with the same two letters, except that the last 2 letters are reversed (a palindrome). `radar` is an example.
9. Do the same thing as number 8 but find all words that start and end with the same 3 letters (the last 3 letters would be in reverse order from the first 3). The word could be 6 or 7 characters long. Hint: I only found 2 words in the list that met this criteria.
10. Find all words that have an aeiou (in order). Each vowel appears only once in the word. I only found 3.
11. Display all words that have 3 consecutive double-letter pairs (like “bookkeeper” has oo, kk, ee)?
12. Find all lowecase [a-z] words that have an apostrophe, but do not end in the letter ’s’.
13. Find all words that consist of only 2 lowercase letters. (The word can only be 2 characters long)
14. Validate an email. The function `fourteen` should receive an string. If it is valid email print out ‘email is valid`, otherwise print out`email is invalid. (email should be replaced with the actual email).
15. Validate an ip. The function `fifteen` should receive an string. If it is valid ip print out ‘ip is valid`, otherwise print out`ip is invalid. (ip should be replaced with the actual ip).
16. Validate an dnumber. The function `sixteen` should receive an string. If it is valid print out ‘dnumber is valid`, otherwise print out`dnumber is invalid`. (dnumber should be replaced with the actual dnumber.) (A`

`dnumber always starts with`d and is followed by 8 digits).

To see if you are doing it correctly you can look below to see if you are getting similar counts for functions 1-13. You could pipe all your function calls in main to the `wc` command.

```
    190     190    1943
     28      28     249
     63      63     598
     17      17     192
    912     912   14593
     25      25     189
     39      39     344
     15      15      90
      2       2      15
      3       3      33
      5       5      62
     31      31     212
    112     112     336
```

This should be uploaded as `week7/assignment.sh` to your github repo.