Perl Regular Expression Excersises

<http://www.ynonperek.com/courses/basic-perl/re-ex.html>

**Part 1**

Write a program that prints only input lines that match the template:

1. Write a regular expression to accept any binary string (only 0s and 1s)
2. A binary string represents an even number if it ends with a 0. Write a regular expression to match an even number represented as a binary string.
3. All lines from ls -l which represent a folder
4. Lines longer than 50 characters
5. Lines that have even length

**Part 2**

Write a program that takes a file with UNIX path names and translate it into windows paths (changing the / to \, and adding drive letter). Sameple Input:

/etc/passwd

/home/bob/bookmarks.xml

/home/bob/vimrc

Sample output:

C:\etc\passwd

C:\home\bob\bookmarks.xml

C:\home\bob\vimrc

**Part 3**

Write a perl program that takes a CSV file with 3 fields: first name, last name, email; and swaps the columns, printing the values: last name, first name, email. A sample input:

Shana,Sargent,shanasargent@isoswitch.com

Witt,Hampton,witthampton@zaphire.com

Morgan,Grant,morgangrant@lotron.com

And the output:

Sargent,Shana,shanasargent@isoswitch.com

Hampton,Witt,witthampton@zaphire.com

Grant,Morgan,morgangrant@lotron.com

[Link To Solutions](https://github.com/ynonp/basic-perl-examples/tree/master/sol/regexp)

**Part 2**

1. Write a program to print every other word in a sentence. So for the input "I Love Writing Perl Code", it should print "I Writing Code"
2. Write a program that takes lines of inputs and prints "Valid" if the line holds a legal perl variable identifier.
3. Write a program to read a file and print lines containing only numbers
4. Write a program that extracts words inside a double quoted strings. For instance, taking the input: he said "Goog Morning" and then went away, should produce the output: Good Morning
5. Write a program that matches a time-of-day string (such as 9:14 am or 11:20 pm).
6. Write a Regular expression to delete duplicate words
7. Write a regular expression to match a valid email address.
8. Write a regular expression to match a valid IP Address (remember that 0.0.0.0 is valid, but 300.0.0.0 is not)
9. Write a regular expression that takes a line and prints only the words that have an odd number of letters
10. Write a program that reads all file names in a directory and prints only the really long ones (longer than 20 characters)
11. Write a program that reads all file names in a directory, and searches for secrets. If a file's extension is .txt, it should read it and print any line in ithe file with the words TOP SECRET somewhere on the line

Write a program that calculates frequency of appearnce for words in text. The program should take a file name from the command line, and, for each word in the file, prints how many times it appears in the text.

<http://www.troubleshooters.com/codecorn/littperl/perlreg.htm>

GROUPING with ( ) (Clinton|Bush)

<https://view.officeapps.live.com/op/view.aspx?src=http%3A%2F%2Fcs.hiram.edu%2F~walkerel%2Fintd388%2FRegularExpressionExercises.doc>

[**Regular Expression exercises**: - Hiram College](http://cs.hiram.edu/~walkerel/intd388/RegularExpressionExercises.doc)

cs.hiram.edu/~walkerel/intd388/**RegularExpressionExercises**.doc

**Regular Expression Practice**

Login to cs.hiram.edu and cd to your bioinformatics directory. Execute the following command to copy the perl file from my directory to yours:

cp /home/webusers/walkerel/intd388/regex.pl regex.pl

Write a regular expression for each of the following exercises. Edit the file regex.pl and replace the expression $regex with your expression. For each question, give the regular expression and the number of words that match. If the number of words that match is 10 or less, give the list of words as well.

1. Find all words in big\_english.txt that have the sequence "cie" somewhere in the word.

2. Find all words in big\_english.txt that contain either the sequence "yes" or the sequence "no".

3. Find all words in big\_english.txt that begin and end with the same vowel (a, e, i, o, or u)..

4. Find all words in big\_english.txt that begin and end with the same 3-letter sequence.

5. Find all words in big\_english.txt that contain at least 3 vowels in a row.

6. Find all words in big\_english.txt that contain no vowels.

7. Find all words in big\_english.txt that contain at least 4 o's (need not be consecutive).

8. Find all words in big\_english.txt that contain 3 copies of the same 3-character sequence.

9. Find all words in big\_english.txt that begin and end with a vowel, but have no vowels in between.

10. Find all words in big\_english.txt that begin and end with a vowel, have no vowels in between, and are at least 5 letters long.

11. Find all words in big\_english.txt that have 3 consecutive double-letter pairs (like "bookkeeper" has oo, kk, ee)

12. Find all words in big\_english.txt that contain both the sequence "yes" and the sequence "no". (Hint: the sequences can occur in either order)

Edit your file to replace big\_english.txt with ecoli.txt.

13. Find all 7-mers in ecoli.txt that contain the sequence “ATG” somewhere in the 7-mer.

14. Find all 7-mers in ecoli.txt that contain the sequence “ATG” preceded by an A or G three nucleotides upstream and followed by a G. This regular expression nicely approximates how real eukaryotic start codons are recognized.

15. Find all 7-mers in ecoli.txt that contain a direct repeat of a two-nucleotide unit (e.g., ACAC, GGGG).

16. Find all 7-mers in ecoli.txt that contain a direct repeat of a two-nucleotide unit separated by a single nucleotide (e.g., ACGAC, GGTGG).

17. Find all 7-mers in ecoli.txt that contain a proline codon followed by a valine codon.

18. Find all 7-mers in ecoli.txt that contain a mirror repeat of a three-nucleotide unit (e.g., ATCCTA).