I427 Lab 3: More Perl, Sept 12 2013

In this lab you’ll get some experience writing basic Perl programs. At the end of the lab period, please submit your work by running the following command:

`/u/djcran/submit lab3`

We will use the submissions to record attendance and to measure the general progress of the class. If you do not finish during class, it is important to complete these exercises on your own, as they are good practice for writing Perl programs.

1. Login to `burrow.soic.indiana.edu` using putty, and create a new folder called `lab3`. You can keep all your files for this lab in this folder.

2. (`re.pl`) In class on Tuesday we discussed regular expressions, which are very useful for processing input from a file or the keyboard. Write a perl program that reads in a string from the user, and then checks and displays whether it is:

   (a) a zip code (i.e. consists of 5 numeric digits)
   (b) a single word (i.e. consists of upper and lower case letters without any spaces)
   (c) a single word with at least one vowel
   (d) a single word that appears to be plural (i.e. ends in either a consonant followed by an “s”, or ends in “es”)
   (e) a proper noun (i.e. a single word that begins with an upper-case letter)
   (f) a telephone number of the form “(xxx)-xxx-xxxx” or “xxx-xxx-xxxx” or “xxx.xxx.xxxx” (where each “x” represents a numeric digit)

Here’s what a sample run of the program might look like:

```
Please enter a string: hippopotamus
zip code? no
single word? yes
single word with vowel? yes
plural? no
proper noun? no
telephone number? no
```

3. (`sieve.pl`) The Sieve of Eratosthenes is an algorithm for finding all of the prime numbers less than some value N. It dates to around the year 200 BC. The algorithm works like this:

   (a) First, create an array containing all of the integers between 0 and N.
   (b) Set the variable k to 2.
(c) Then, go through the list. If a number is divisible by \( k \) but not equal to \( k \), then set that array entry to 0. Otherwise leave it unchanged. (You can check whether one number is evenly divisible by another number using Perl’s modulus (remainder) operator, \( \% \). For example, \( 5 \% 3 \) evaluates to 2, because the remainder of dividing 5 by 3 is 2.)

(d) Now add 1 to \( k \).

(e) If \( k \) is less than or equal to \( N \), go to step (c). Otherwise go on to step (f).

(f) Finally, print out all of the non-zero entries remaining in the array. These should be the prime numbers.

Write a Perl program that implements the Sieve, with \( N=100 \). Does it work well with \( N=1000 \)? How about \( N=1000000 \)?

Once you have the above algorithm working, try some changes to make it run faster.

— You can change step (e) to go to step (c) only if \( k^2 \) is less than \( N \).
— Instead of adding 1 to \( k \) in step (d), instead set \( k \) to the next non-zero number left in the array that is greater than \( k \).
— In step (c), instead of looping through every element and testing whether it is divisible by \( k \) or not, you can instead simply set all array indices that are multiples of \( k \) to 0. (For example, if \( k = 2 \), set the elements at indices 2, 4, 6, 8, etc. to 0).