**Program Question 1.**

This program reads in a file that represents an address book. The address book lines have the following format:

FirstName|LastName|Street Address|City|State Code|Zip Code|Phone|Email

This program will read in the input file addresses.txt (link to be provided on the web site) and syntax-check each field for correctness.

Your program should use a series of regular expressions to verify that the data meets the formatting rules. The rules are:

1. The first and last name should both start with a capital letter. There should be only letters, spaces and ' allowed in a name.
2. The street address starts with one or more digits, then a space, then a street name which can contain letters only, then a space then the code for the type of street (allowable types are St. Rd. Dr. Cir. Ln. and Blvd. for simplicity). We are only going to test for one-word street names – things like 123 Cedar Lake Dr. do not need to be accounted for.
3. The city is simply words and spaces where the words have only letters, no numbers.
4. State code is always two capital letters
5. Zip code is either five numbers or five numbers followed by a dash and four more numbers.
6. The phone number with area code has this format: (XXX) XXX-XXXX
7. The email address should contain at least an @ sign and at least one . after the @ sign.

(The real pattern for a valid email is famously complicated; this is a major simplification)

As each record (line of the file) is read in, check the input. If it is valid, print a success message that says “The entry for John Smith is valid.”. If it is invalid, inform the user of the problem(s: “The entry for John Smith has the following problems: Invalid Street Address, Invalid Phone Number.”

Finally, you should produce two output files: valid.txt and invalid.txt, which should contain the valid and invalid entries, respectively.

**Program Question 2. Word and Letter Counter**

You will be given a text file as input (link provided on web site). Your goal is to output the following:

1. a list of words sorted by the number of times they appear in the text file (most common word first) along with the number of times they have appeared.
2. a list of letters sorted by the number of times they appear in the file (most common letter first) along with the number of times each has appeared. We're only interested in letters, no punctuation or spacing. A capital B and a lowercase b should be counted the same – they're both b's. In other words, I don't want a separate count for B and b – just one for all b's.

So, for each line of input, you'll be doing two simultaneous counts.

Tips and Tricks

1. Remember the trick we learned in class about using hashes for counts.

2. split //, $line splits a line on the empty string. See perldoc -f split for more.

3. You'll want to sort a hash based on the values of the hash (most common word first, etc...). This can be accomplished by defining a custom sort.

sort {CUSTOM SORT DEFINITION GOES HERE} (keys %hash);

When you use the sort function, you get access to two special variables: $a and $b. These variables represent “the two things currently being compared” from the list to be sorted. You can use these variables inside your custom sort definition. The end result of your sort is that the left item is first in the sort, the right item is first in the sort, or the left and right items are equivalent.

To do this, you need to use the “three way comparison” operators: cmp and <=> which are documented in “perldoc perlop”

In the example above, our list to be sorted is the output of (keys %hash). So, $a and $b are going to contain the names of keys within the hash. Since our goal is to sort by the hash values (e.g. the number of times each word has appeared), you can use something like this as

your sort definition:

my @sortedkeys = sort {$hash{$a} cmp $hash{$b}} (keys %hash);

Now, you'll get the list of keys in the order of their sorted values. Tricky, but very handy! This isn't quite the code needed to do what is asked in this program, but it will get you close enough to figure it out on your own.