Homework 9

All questions should read their input from stdin and write their output to stdout.

1. Rework question 4 from homework 8, but with the following additional specifications:
    a. case should be ignored. For example, "The" and "the" should be treated as the same word. You can use the lc (lower case) function to convert a word to all lowercase.
    b. a word may be preceded or succeeded by punctuation, such as a comma, a period, or a # sign. You need to strip any leading or trailing punctuation. You may assume that a word starts with an alphanumeric character (alphabetic character, digit from 0-9, or _) and ends with the first non-alphanumeric character that is encountered. For example:
        #include, ➔ include
        &!Brad!&3 ➔ Brad
        $dict{$word}\n"; ➔ dict
        3 ➔ 3
        { ➔ discarded because it does not contain an embedded word
        for(i=0;i<10;i++) ➔ for
2. Write a script named **times.pl** that reads a file of data that contains race results of the form h:mm:ss, where h is hours, mm is minutes, and ss is seconds and perform the following two actions:
    a. Calculate and print the average of all the times in the file. The printf statement that will force a leading 0 into the minutes or seconds position if the average minutes or seconds is a single digit is:
        printf("average time = %d:%02d:%02d\n")
        The 0 in %02d says to pad the field with leading 0's if there are not enough digits to fill the field.
    b. Convert each string from the form "h:mm:ss" to the form "h hours, mm minutes, and ss seconds". Use a substitution pattern to do the replacement inline and write out the converted file.

    Hours is a single digit while minutes and seconds must be 2 digits. You may assume that the times have been correctly entered. As an example, if your file is:

    1:23:06
    3:10:45

    then your output should be:

    average time = 2:16:55
    1 hours, 23 minutes, and 06 seconds
    3 hours, 10 minutes, and 45 seconds

3. Write a Perl program named **center.pl** that searches for all header tags of the form <h1>, <h2>, or <h3> that start at the beginning of a line in an html file. Place the tags and around all such headers. The headers may span more than one line in the file. For example, your program should replace:
        <h1> An Introduction to Perl
        and Python 101 </h1>.

with:

<h1> An Introduction to Perl
and Python 101 </h1> .

4. Write a Perl script named **sections.pl** that finds and extracts all section headers in a file and prints their content one per line. Section headers are of the form \section{text}, \subsection{text}, \subsubsection{text}, \subsubsubsection{text}, or \subsubsubsubsection{text} and they must start at the beginning of a line, although they may have leading whitespace characters. I want the text in the section headers to be cleaned up as follows:
    a. You should remove (i.e., soak up) any leading whitespace between the "{" and "text" and any trailing whitespace between "text" and the "}" before printing text.
    b. You should remove any newline characters and replace them with spaces.
    c. You should remove multiple whitespace characters between words and replace them with a single space (note that you can combine steps b and c into one step if you're clever).

    For example, given:
    \section{Project Description}
    …
    \subsubsubsection{ Brad's Amazing
     Project and
       It's Aftermath }

    your script should print

    Project Description
    Brad's Amazing Project and It's Aftermath

5. Go to http://web.eecs.utk.edu/~bvz/teaching/cs140fa08/labs/lab2/ and look at the description of pgm files in part 3. Then write a script named **rot90.pl** that reads a pgm file and writes out a 90-degree rotation. Note that this will turn an input pgm file with *r* rows and *c* columns into an output pgm with *c* rows and *r* columns. Here are a few tips/comments:
    a. Ignore the references to C++/C (e.g., ignore references to malloc) in the problem description. I just want you to become familiar with the format of pgm files.
    b. The sample pgm files are now in /home/bvz/courses/140/fall-2008/labs/lab2, not the directory listed on the web-site.
    c. If you use split to split the rows of pixels into individual pixel fields, remember to first strip away any leading whitespace, otherwise you may encounter a problem with leading empty fields. You can strip away leading whitespace using the sponge pattern (\s*) discussed in class. For example, if $line contains the line you just read, then the Perl command:
        $line =~ s/^\s+//;
    will remove leading spaces. Notice that if the pattern fails because there is no leading spaces, then the string in $line will remain unchanged, which is just fine.