Homework 6

For your Blackboard submission, please submit 5 separate .pl files, one for each problem. We will test your files with our own data. The test files shown in the problems are provided as blackboard attachments to this assignment.

1. Create a Perl file named `stddev.pl` that computes and prints the mean, sample variance, and sample standard deviation of a set of numbers. The numbers should be read from stdin. For example:

```
bvz% perl stddev.pl < dataset1.txt
average = -599.43
sample variance = 3238544.21
sample standard deviation = 1799.60
```

   You should use the unbiased sample variance formula, which is:

$$Sample\ variance = \frac{1}{n-1}\left(\sum_{i=1}^{n}(y_i - \bar{y})^2\right)$$

   The sample standard deviation is the square root of the sample variance.

2. You are asked to verify some form data from a web-site. For convenience, the data has been stored in a text file, which you will read from STDIN and verify. Create a Perl file named `verify.pl` that performs the following checks on the data and prints a suitable error message if the check fails (you may assume that the data is presented in the same order as the checks). Each check must be accomplished exclusively using a regular expression:
   a. Check that the room type is one of "double', 'queen', 'king', or 'suite'
   b. Check that the check-in date has the form mm-dd-yyyy with the month, day, and year all being integers, the year starting with the prefix 20, and the month and day having either one or two digits.
   c. Check that the guest's name is 30 characters or less and that the guest's name is all alphanumeric characters.
   d. Check that the guest's preferred club member identifier is a string that starts with either 'pl' or 'gl' and then has an optional dash (-) and a four digit sequence from 1000-9999. For example, `pl-3868` and `gl1586` are fine, while `foo-3858` and `pl-0138` should be flagged as errors.
   e. Check that the phone number has the following format
      i. a leading '011' country code, indicating that we'll be calling from the US
      ii. a 1-3 digit country code for the country we're calling to
      iii. a 2 digit city code for the city we're calling to
      iv. a code of the form ddd-dddd for the local phone number (3 digits, a dash, then 4 digits)

      Legitimate separators for the groups are a dash (-) or a space, except for the local phone number, which must have a dash. The 011 country code may optionally be enclosed in parentheses. A couple example phone numbers would be `(011) 42 55 363-6858` and `011-3-36-386-3969`.

   A file contains data for only one reservation, and each of the five data elements will be on a separate line. For example:

```
king
```

```
3-2-2010
Brad Vander Zanden
gl-1068
(011) 42-55-363-6858
```

Your script should ensure that there is no extraneous information on a line. For example, if someone entered "suiteking" or "kingsuite" instead of "king", then the entry should be flagged as an error. The file `regexp1.txt` is a sample file with good data and `regexp2.txt` is a sample file in which each entry is invalid. You should make sure that you modify the data in each file so that you exhaustively check your program. The data in the two files does not come close to exhausting both the types of good and bad data.

3. Write a Perl script named `time.pl` that repeatedly prompts a user to enter military times in the format hh:mm:ss where hh is an hour from 0 to 23, mm is a minute from 0 to 59, and ss is a second from 0 to 59. The program should output the times in standard am/pm format. For example, 17:23:49 should be output as 5:23:49pm. You may assume that the time is entered correctly (i.e., no error checking is required). The split command may be used to split this string into its constituent parts, using : as a delimiter.

4. Write a program named `word_count.pl` to count the frequency of words in standard input (STDIN) and then print the words and their frequency in any order. For example, given the file fox.txt:

```
The quick brown fox jumped
the brown fence before dinner and then
met joey fox for dinner.
```

and the command:

```
perl word_count.pl < fox.txt
```

your script might output:

```
quick: 1
The: 1
the: 1
joey: 1
fox: 2
dinner: 1
dinner.: 1
… rest of output …
```

For this exercise, you:

1. should treat case as significant ("The" and "the" count as different words) and punctuated words as different ("dinner" and "dinner." both count as words).
2. should use whitespace as a delimiter and you should be prepared for the case where there are multiple spaces between words. You do not have to worry about either leading spaces at the beginning of the line or trailing spaces at the end of the line
3. **must use a hash table (i.e., an associative array) to do your word counting.**

5. Write a Perl script named `grades.pl` that reads student records from STDIN and computes their final grade in the course. Student records will have the format:

   lastname,firstname,assessmentType,score

   and there will be one student record per line. The delimiter between fields is a comma, and you may assume that there are no spaces before or after the commas. The assessmentType is one of "hw", "midterm", and "final". For example:

```
vander zanden,brad,hw,75
jones,mark,midterm,58
mouse,mickey,hw,83
jones,mark,final,67
squirrel,chip,hw,95
hound,smiley,hw,85
vander zanden,brad,midterm,85
vander zanden,brad,final,95
jones,mark,hw,67
squirrel,chip,final,100
squirrel,chip,midterm,65
hound,smiley,midterm,97
mouse,mickey,final,85
hound,smiley,final,95
mouse,mickey,midterm,57
```

   Your script will need to accumulate information about each student. Therefore it **must** create an anonymous hash to store information about each student (my hashes contained keys for firstname, hw, midterm, and final). The reference to this hash object should then be stored in a named hash table, such as a hash table named `grades`, with the key being the student's last name. Once you've read through all the input, your script should iterate through the named hash table and calculate the final score for each student as:

   final_score = 0.2*hw + 0.3 * midterm + 0.5 * final

   You should then print out each student's grade using the format:

   last name, first name final_score

   For example:

```
% perl grades.pl < grades.txt
squirrel, chip 88.5
jones, mark 64.3
mouse, mickey 76.2
vander zanden, brad 88
hound, smiley 93.6
```

   I realize that if you are clever, you could write this script using only a named a hash table (you could simply accumulate the score as you went along). However, I simplified

this problem from having multiple homework and midterm assignments, where you would have been forced to maintain a running total of a student's homework and midterm grades, as well as the number of homework assignments and midterms. The point of this problem is to give you practice with creating records in Perl by using anonymous hashes, and then storing references to these hashes in another named data structure.