Programming in Perl, Fall 97
Homework 1
Due : Friday October 31 at midnight
Weight : 10% of course grade
Instructor : Louis Ziantz

This assignment consists of a series of small Perl programs. It should be submitted according to the *Submission Guidelines*. Don't forget to comment your program, providing a general description of the Perl script along with inline comments for important or complex portions of code. Please get started on the assignment as soon as possible.

Problem 1 Write a Perl script called palindrome.pl that detects if a given string input by the user is a *palindrome*. A palindrome is a word or phrase that is reads the same both forward and backward. For example, 'radar', '1234321', and 'able was I ere I saw elba'. Capitalization is immaterial, that is, words like 'Anna' and 'aBcbA' should be considered palindromes. However, spaces and punctuation should not be ignored, so 'A man a plan a canal Panama' is not a palindrome for this exercise (this simplifies the problem). The user should be prompted to enter a string. After the data is entered, the script should indicate whether or not the string is a palindrome while including the string in the output. For example, if the user enters `radar`, the script should print `The string 'radar' is a palindrome.`

Problem 2 Write a Perl script that functions like the UNIX utility `sort` applied to STDIN. Running sort.pl should allow the user to enter a series of words (or other lines of text) each followed by newlines, with the entire list terminated by a Ctrl-D. The script should read the lines into an array, sort them based on ASCII ordering, and then print the sorted array. If five lines are read in, five lines should be written out. The array should NOT be written out so that a blank is prepended to any of the lines. Thus, if given a file words.dat, we can run `sort.pl < words.dat > sorted.words.dat` to store the sorted version of words.dat in sorted.words.dat.

Problem 3 Write a Perl script employee.pl that computes the weekly pay for an employee. The script should first prompt for an employee name. If the user just hits RETURN for the name, the program should terminate. As long as we have a valid employee name, the script should prompt for a paycode, read in the numeric paycode, prompt for additional data based on the paycode, read in that data, compute and print pay appropriately, and prompt for and read in the name of the next employee.

The paycodes are as follows:

1 – hourly worker
   - *data needed for pay computation:* hours worked and hourly wage
   - *pay computed as:* hourly wage * hours worked if hours worked <= 40; hourly wage * 40 + (1.5 * hourly wage * hours beyond 40 worked) otherwise

2 – commission worker
   - *data needed for pay computation:* gross weekly sales
   - *pay computed as:* $250 + 5.7% of gross weekly sales

3 – pieceworker
   - *data needed for pay computation:* pay per item and number of items made
   - *pay computed as:* pay per item * number of items made

For example, if `Bob Smith` is entered as the employee name, and `1` is entered as the paycode (hourly worker), then the script should prompt for hourly wage (suppose `8.00` is entered) and hours worked (suppose `50` is entered). The script would then print `The weekly pay for Bob Smith is $440.` and prompt for the next employee, exiting if just a RETURN is entered.

Problem 4 Write a Perl script that simulates the rolling of two dice. The script dice.pl should first initialize (seed) the random number generator by calling `srand;`. You should then prompt the user for the number of times the dice should be rolled and read that input. Because the `rand(6)` function produces a floating point value on the interval [0,6), the roll of a single six-sided die may be simulated using: `int(rand(6)) + 1`. You then just need to compute the sum of two individual die rolls to simulate rolling two dice. Use an array to tally the numbers of times each possible sum appears. When the number of rolls specified by the user are complete, output a summary. For each sum from 2 to 12, print the sum, a colon, and the number of times it was rolled followed by a newline.

Problem 5 Write a Perl script called seven.pl that prints out *The Seven Lectures of Perl.* You will need to initialize an array `@num_to_day` with a list of `'First'`, `'Second'`, on up to, `'Seventh'`. You can then use `@num_to_day` to initialize hash `%day_to_num` using a hash slice so that `%day_to_num` has key-value pairs ('First',0), ('Second', 1), and so forth to, ('Seventh', 6). You should also create a hash `%seven_days` with the following keys and values:

'First'	'Literals and scalars'
'Second'	'Arrays and control structures'
'Third'	'Hashes and regular expressions'
'Fourth'	'Whatever we fit in after the test'
'Fifth'	'Simple CGI programming'
'Sixth'	'Whatever we fit in before the exam'
'Seventh'	'What the exam actually looked like'

For each `$key` in the hash `%seven_days`,

- print `"By the $key day of Perl class, we learned about...\n"`
- use `$key` on the hash `%day_to_num` to get a number, call it `$bound`
- iterate from `$bound` downto 0 (call the loop variable `$i`),
  - use `@num_to_day` and `$i` to create an appropriate `$daykey` into the hash `%seven_days`
  - print a tab followed by the result of accessing `%seven_days` with `$daykey` followed by a newline

So, if `$key` is `'Third'`, we get the following:

```
By the Third day of Perl class, we learned about...
        Hashes and regular expressions
        Arrays and control structures
        Literals and scalars
```

Since we are iterating over a hash, will we be printing the "By the $key day of..." lines out in order from First to Second to ... Seventh? Think about it.