Homework #1: CMPT-413

Distributed on Mon, Jan 17; due on Mon, Jan 24

Anoop Sarkar – anoop@cs.sfu.ca

(1) Perl programming warm-up exercises: For this question, use the text file called `hw1.txt` as the input to each of the sub-parts to this question. The location of this file is given on the course web page (look under the *Assignments* section). Each sub-part indicates the filename that you *must* use for your Perl program, e.g. (`your_program.pl`). Each program should run from the command line as follows:
`% perl your_program.pl hw1.txt` ⟨*cr*⟩

Read the perl manual pages by running `man perl` which provides an index to other perl manual pages. The important ones to read right away are `man perlfunc`, `man perlop` and `man perldata`.

Here is an example perl program to count the number of words in a file:

```
use strict;
my $line;
my $total = 0;
while ($line = <>) {
  chomp($line);
  my @words = split(' ', $line);
  next if ($#words < 0);
  $total += $#words +1;
}
print "$total\n";
```

Save this as a file `wc.pl` and run `perl wc.pl hw1.txt`. Compare your output with the standard Unix command `wc` which also reports the number of words. The command `use strict;` ensures that a lot of errors are caught at compile-time.

Important: In this and future homeworks, use the Unix command `make` to run all your submitted programs. For this homework, use the `makefile` provided in the homework directory (see the course web page). Using the provided `makefile`, once you have written all the programs below, you can run all of them by running `make test`. For this homework, the output of `make test` is provided as the file `log` in the homework directory.

a. (`wc_lines.pl`) Modify the above program to report the number of lines in addition to the number of words. Run it on `hw1.txt`. You should get 20 lines.

b. (`ln.pl`) Write a Perl program to read in the text file and print each line to standard output with the line number of each line as a prefix. The line numbers should start from 1. e.g. it would take each line of text from the input:

```
some text here
some more text here
```

and print each line with it's line number:

```
1 some text here
2 some more text here
```

Also print out the number of lines to STDERR. e.g. if the file had 20 lines then print to STDERR: `number of lines=20`

c. (`stripln.pl`) Write a Perl program that takes the text file and strips away any initial positive integer followed by a space if it exists, e.g. it would take a line of text:

```
1 some text here
2 some more text here
```

and print

```
some text here
some more text here
```

d. (`shuffleLines.pl`) Write a Perl program to randomly shuffle the lines in the text file. Use the following function `fisher_yates_shuffle` which randomly shuffles an array in place:

```
sub fisher_yates_shuffle {
    my ($array) = @_;
    for (my $i = @$array; --$i; ) {
        my $j = int(rand($i+1));
        next if ($i == $j);
        @$array[$i,$j] = @$array[$j,$i];
        # note that: @$array[$j,$i] equals ($array[$j],$array[$i])
    }
}
```

To use the above function, you have to pass it a reference to an array which is then shuffled in place. e.g.

```
@x = (1,23,34,22,342,11,23);
fisher_yates_shuffle(\@x);
print "@x\n";
```

This produces the result:

```
11 1 34 23 342 22 23
```

e. (`shuffleWords.pl`) Use the function `fisher_yates_shuffle`, but this time use it in a Perl program to randomly shuffle the words in each input line. Each line of text should appear in the same order as the input but the words in each line should be randomly permuted.

f. (`removeDuplicates.pl`) Write a Perl program to eliminate duplicate lines from the input file. Provide the number of lines that remain after all duplicate lines are eliminated from `hw1.txt`. You should get 10 unique lines.