COMP[29]041 16s2 (http://www.cse.unsw.edu.au/~cs2041/16s2/) Software Construction (http://www.cse.unsw.edu.au/~cs20

More Perl

Aims

Get used to more Perl features.

Assessment

Submission: `give cs2041 lab06 whale.pl all_whales.pl prereq.pl [recursive_prereq.pl digits.py echon.py simple_tail.py whale.py all_whales.`
Deadline: either during the lab, or Monday 5 September 11:59pm (midnight)
Assessment: Make sure that you are familiar with the lab assessment criteria (lab/assessment.html).

Background

We have covered only a small amount of Perl in lectures. In fact, to cover the whole language in detail would take a whole semester, so we're going to rel you finding out about the language yourself in tutes, labs and assignments. A good place to start is the Perl documentation & tutorial links on the class ho page For example you might find these useful:

- Perl language syntax (http://search.cpan.org/dist/perl/pod/perlsyn.pod)
- Perl functions (http://search.cpan.org/dist/perl/pod/perlsub.pod)
- Perl operators (http://search.cpan.org/dist/perl/pod/perlop.pod)

Input Data

A biologist needs your help counting migrating whales. As pods of migrating whales as they swim past the biologist's boat, the biologist types the number species of whales in the pod
Here is an example of the input the biologist will provide

```
 15 humpback
2 orca
2 sperm whale
19 beluga
2 humpback
2 sperm whale
4 orca
4 pygmy right whale
7 humpback
1 orca
```

Counting One Whale Species

Write a Perl script `whale.pl` which given a whale name as a command line argument, reads whale observations in the above format until the end-of-inpu reached and then prints the number of pods of the specified whale and the total numbers of whales in those pods. For example:

```
$ ./whale.pl 'sperm whale'
 15 humpback
2 orca
2 sperm whale
19 beluga
2 humpback
2 sperm whale
4 orca
4 pygmy right whale
7 humpback
1 orca
Ctrl-d
sperm whale observations: 2 pods, 4 individuals
```

```
$ ./whale.pl "cuvier's beaked whale"
 3 orca
673 beluga
2 sperm whale
123 pilot whale
2 cuvier's beaked whale
19 beluga
2 north atlantic right whale
2 humpback
5 cuvier's beaked whale
2 sperm whale
1 north atlantic right whale
1119 beluga
7 orca
4 pygmy right whale
7 humpback
1 false orca
3 beluga
4 pilot whale
1 orca
Ctrl-d
cuvier's beaked whale observations: 2 pods, 7 individuals
```

As usual use `autotest` to assist in testing your code and push your work to `gitlab.cse.unsw.edu.au` every time you make some progress with it.

```
$ ~cs2041/bin/autotest lab06 whale.pl
...
$ git add whale.pl
$ git commit -a -m "whales are cool just like Andrew"
...
$ git push
...
```

Sample Perl solution

```
#!/usr/bin/perl -w

die "Usage: $0 <whale species>\n" if @ARGV != 1;
$target_species = $ARGV[0];

$n_pods = 0;
$n_individuals = 0;
while ($line = <STDIN>) {
    if ($line =~ /(\d+)\s*(.+)$/) {
        $count = $1;
        $species = $2;
        if ($species eq $target_species) {
            $n_pods++;
            $n_individuals += $count;
        }
    }
}
print "$target_species observations: $n_pods pods, $n_individuals individuals\n";
```

Sample Python solution

```
#!/usr/bin/python

import re, sys

if len(sys.argv) != 2:
    sys.stderr.write("Usage: %s <whale species>\n" % sys.argv[0])
    sys.exit(1)
target_species = sys.argv[1]

n_pods = 0
n_individuals = 0
for line in sys.stdin:
    m = re.search(r'(\d+)\s*(.+)$', line)
    if m:
        count = m.group(1)
        species = m.group(2)
        if species == target_species:
            n_pods += 1
            n_individuals += int(count)
print("%s observations: %d pods, %d individuals" % (target_species, n_pods, n_individuals))
```

Counting All Whale Species

Write a Perl scripts `all_whales.pl` that reads whale obsevatiions in the above format until the end-of-input is reached and for all species of whale observ prints the number of pods of the specified whale and the total numbers of whales in those pods. For example:

```
$ ./all_whales.pl
 15 humpback
2 orcas
2 sperm whales
19 belugas
2 humpbacks
2 sperm  whales
4 Orcas
4  pygmy right whale
7 humpbacks
1 orca
3 ORCAS
673 belugas
2 sperm whales
123 pilot whale
2 cuvier's beaked whales
19 beluga
2 north atlantic right whale
2 humpbacks
5 cuvier's BEAKED whale
2 Sperm Whales
1 north atlantic right whales
1119            belugas
7 orcas
4 pygmy right whales
7 humpbacks
1 False Orca
3 belugas
4 pilot whaleS
1 orca
Ctrl-d
beluga observations: 5 pods, 1833 individuals
cuvier's beaked whale observations: 2 pods, 7 individuals
false orca observations: 1 pods, 1 individuals
humpback observations: 5 pods, 33 individuals
north atlantic right whale observations: 2 pods, 3 individuals
orca observations: 6 pods, 18 individuals
pilot whale observations: 2 pods, 127 individuals
pygmy right whale observations: 2 pods, 8 individuals
sperm whale observations: 4 pods, 8 individuals
```

The whales should be listed in alphabetical order.

All whale names should be converted to lower case.

All whale names should be converted from plural to singular - assume this can be done safely by deleting a trailing 's' if it is present.

Any extra white space should be ignored.

You can make no assumptions about possible whale names.

No mention of particular whale names can appear in your program.

As usual:

```
$ ~cs2041/bin/autotest lab06 all_whales.pl
...
$ git add all_whales.pl
$ git commit -a -m "whales & perl what could be better"
```

Sample Perl solution

```
#!/usr/bin/perl -w

while ($line = <STDIN>) {
    $line = lc $line;     # convert line to lower case
    $line =~ s/\s+$//;    # remove trailing white space
    $line =~ s/s?$//;     # change to singular
    $line =~ s/\s+/ /g;   # convert sequential white-space charcaters to a single space

    if ($line =~ /(\d+)\s*(.+)\s*$/) {
        $count = $1;
        $species = $2;
        $n_pods{$species}++;
        $n_individuals{$species} += $count;
    } else {
        print "Sorry couldn't parse: $line\n";
    }
}

foreach $species (sort keys %n_pods) {
    print "$species observations: $n_pods{$species} pods, $n_individuals{$species} individuals\n";
}
```

Sample Python solution

```
#!/usr/bin/python

import re, sys

n_pods = {}
n_individuals = {}
for line in sys.stdin:
    line = line.lower()              # convert line to lower case
    line = line.strip()              # remove surrounding white space
    line = re.sub(r's?$', '', line)  # change to singular
    line = re.sub(r'\s+', ' ', line) # convert sequential white-space charcaters to a single space
    m = re.search(r'(\d+)\s*(.*)$', line)
    if m:
        count = int(m.group(1))
        species = m.group(2)
        if species in n_pods:
            n_pods[species] += 1
            n_individuals[species] += count
        else:
            n_pods[species] = 1
            n_individuals[species] = count
    else:
        print("Sorry couldn't parse: %s" % line)

for species in sorted(n_pods):
    print("%s observations: %d pods, %d individuals" % (species, n_pods[species], n_individuals[species]))
```

Prerequisites

Write a Perl script which prints courses which can be used to meet prerequisite requirements for a UNSW course. For example:

```
$ ./prereq.pl COMP2041
COMP1917
COMP1921
$ ./prereq.pl COMP9041
COMP9021
$ ./prereq.pl COMP9242
COMP3231
COMP3891
COMP9201
COMP9283
$ ./prereq.pl HESC3641
HESC2501
```

Your script must download the UNSW handbook web pages and extract the information from them when it is run.

You should print the courses in alphabetic order.

Hints

The UNSW handbook uses separate web pages for undergraduate and postgraduate handbook and you may need to extract prerequisites from either or b

A simple way (but not the best way) to access a web page from Perl is like this:

```
$url = "http://www.handbook.unsw.edu.au/postgraduate/courses/2015/COMP9041.html";
open F, "wget -q -O- $url|" or die;
while ($line = <F>) {
    print $line;
}
```

You'll have to make some assumptions about the handbook pages.

It easy in Perl to skip lines until you find one specifying prerequisites.

It easy in Perl to remove part of a line.

It easy in Perl to remove HTML tags.

```
$ ~cs2041/bin/autotest lab06 prereq.pl
...
$ git add prereq.pl
$ git commit -a -m "web scraping isn't as cool as whales"
```

Sample solution for prereq.pl

```
#!/usr/bin/perl -w

$year = 2016;
$url_base = "http://www.handbook.unsw.edu.au/";
$url_ugrad = "$url_base/undergraduate/courses/$year";
$url_pgrad = "$url_base/postgraduate/courses/$year";

foreach $course (@ARGV) {
    open my $f, '-|', "wget -q -O- $url_ugrad/$course.html $url_pgrad/$course.html" or die;
    while ($line = <$f>) {
        # look for pre-requisite line handling varying format used  in handbook pages
        if ($line =~ /pre.?(requisite)?(.?:|\s+[A-Z]{4}\d{4})/i) {
            $line = uc $line;
            $line =~ s/<[^>]*>/ /g;
            $line =~ s/EXCLU.*/ /i;
            my @courses = $line =~ /([A-Z]{4}\d{4})/g;
            push @prereqs, @courses;
        }
    }
    close $f;
}
foreach $course (sort @prereqs) {
    print "$course\n";
}
```

Sample Python solution for prereq.pl

```
#!/usr/bin/python
import sys, subprocess, re

year = 2016
url_base = "http://www.handbook.unsw.edu.au"

prereqs = []
for course in sys.argv:
    ugrad_url = "%s/undergraduate/courses/%d/%s.html" % (url_base, year, course)
    pgrad_url = "%s/postgraduate/courses/%d/%s.html" % (url_base, year, course)

    # there are python libraries which provide a  better way to fetch web pages
    # subprocess.check_output is also usually better, but here we may get a non-zero exit
    # if either pgrad or ugrad page doesn't exist
    webpage = subprocess.Popen(["wget","-q","-O-", ugrad_url, pgrad_url], stdout=subprocess.PIPE).communicate()[0]
    webpage = webpage.decode("utf-8") # guess the encoding
    for line in webpage.split('\n'):
        # look for pre-requisite line handling varying format used  in handbook pages
        if re.search(r'pre.?(requisite)?(.?:|\s+[A-Z]{4}\d{4})', line, re.I):
            line = line.upper()
            line = re.sub(r'<[^>]*>', '', line)
            line = re.sub(r'EXCLU.*', '', line)
            for word in re.findall(r'[A-Z]{4}\d{4}', line):
                prereqs.append(word)

for course in sorted(prereqs):
    print(course)
```

Challenge Question

Add a -r (for recursive) flag to `prereq.pl` so it also also lists all courses which can be used to meet a prequisite requirement of any course can be used t prerequisite requirements and so on.

Call your new program `recursive_prereq.pl` so

For example:

```
$ ./recursive_prereq.pl -r COMP9243
COMP1911
COMP1917
COMP1921
COMP1927
COMP2121
COMP3231
COMP3331
COMP3891
COMP3931
COMP9021
COMP9024
COMP9032
COMP9201
COMP9283
COMP9331
ELEC1111
ELEC1112
ELEC2141
ELEC2142
MTRN2500
MTRN3500
TELE3018
```

You may find courses listed as prerequisites which are no longer offered and are not in the current handbook - you should include them but you don't nee find their prerequisites (don't look up old handbooks).

Beware infinite loops!

Sample solution for recursive_prereq.pl

```
#!/usr/bin/perl -w

$debug = 0;
$recursive = 0;
$year = 1900 + (localtime(time))[5];
$url_base = "http://www.handbook.unsw.edu.au/";
$url_ugrad = "$url_base/undergraduate/courses/$year";
$url_pgrad = "$url_base/postgraduate/courses/$year";

sub prereq {
    my ($course) = @_;
    print STDERR "prereq($course) $url_ugrad/$course.html $url_pgrad/$course.html\n" if $debug;
    open my $f, '-|', "wget -q -O- $url_ugrad/$course.html $url_pgrad/$course.html" or die;
    my (@prereqs, $line);
    while ($line = <$f>) {
        # look for pre-requisite line (note format used varies in handbook pages)
        next if $line !~ /pre.?(requisite)?(.?:|\s+[A-Z]{4}\d{4})/i;
        $line = uc $line;
        $line =~ s/<[^>]*>/ /g;
        $line =~ s/EXCLU.*/ /i;
        my @courses = $line =~ /([A-Z]{4}\d{4})/g;
        push @prereqs, @courses;
    }
    print STDERR "prereq($course) -> @prereqs\n" if $debug;
    foreach my $course (@prereqs) {
        prereq($course) if !$prereqs{$course}++ && $recursive;
    }
    close $f;
}

foreach $arg (@ARGV) {
    if ($arg eq "-r") {
        $recursive = 1;
        next;
    } else {
        prereq($arg);
    }
}
print "$_\n" foreach sort keys %prereqs;
```

Sample Python solution for recursive_prereq.pl

```
#!/usr/bin/python

import sys, time, subprocess, re

year = time.localtime().tm_year
url_base = "http://www.handbook.unsw.edu.au"
debug = 0

def prereq(course):
    if debug: print("prereq(%s)" % course)
    ugrad_url = "%s/undergraduate/courses/%d/%s.html" % (url_base, year, course)
    pgrad_url = "%s/postgraduate/courses/%d/%s.html" % (url_base, year, course)

    # there are python libraries which provide a  better way to fetch web pages
    # subprocess.check_output is also usually better, but here we may get a non-zero exit
    # if either pgrad or ugrad page doesn't exist
    webpage = subprocess.Popen(["wget","-q","-O-", ugrad_url, pgrad_url], stdout=subprocess.PIPE).communicate()[0]
    webpage = webpage.decode("utf-8") # assume hansdbook uses 'utf-8'
    listed_prereqs = []
    for line in webpage.split('\n'):
        # look for pre-requisite line handling varying format used  in handbook pages
        if re.search(r'pre.?(requisite)?(.?:|\s+[A-Z]{4}\d{4})', line, re.I):
            line = line.upper()
            line = re.sub(r'<[^>]*>', '', line)
            line = re.sub(r'EXCLU.*', '', line)
            for word in re.findall(r'[A-Z]{4}\d{4}', line):
                listed_prereqs.append(word)
    for course in listed_prereqs:
        if course not in prereqs:
            prereqs[course] = 1
            if recursive:
                prereq(course)

if __name__== "__main__":
    recursive = 0
    prereqs = {}
    for arg in sys.argv[1:]:
        if arg == "-r":
            recursive = 1
        else:
            prereq(arg)
    for course in sorted(prereqs.keys()):
        print(course)
```

Python challenge questions

Students coping well with Perl are encouraged to try picking up some Python as well.

The brief introduction to Python in lectures will come later.

The example Python scripts (code/python/code_examples.html) and links to external Python resources should help - but you may need more info - Googl
your friend.

Challenge Exercise: Mapping Digits in Python

Write a Python script `digits.py` that reads from standard input and writes to standard output mapping all digit characters whose values are less than 5 i
character ' < ' and all digit characters whose values are greater than 5 into the character ' > '. The digit character '5' should be left unchanged.

Sample Input Data	**Corresponding Output**
`1 234 5 678 9`	`< <<< 5 >>> >`

`I can think of 100's`
`of other things I'd rather`
`be doing than these 3 questions`
	`I can think of <<<'s`
	`of other things I'd rather`
	`be doing than these < questions`

`A line with lots of numbers:`
`123456789123456789123456789`
`A line with all zeroes`
`000000000000000000000000000`
`A line with blanks at the end`
`1 2 3`
	`A line with lots of numbers:`
	`<<<<5>>>><<<<5>>>><<<<5>>>>`
	`A line with all zeroes`
	`<<<<<<<<<<<<<<<<<<<<<<<<<<<`
	`A line with blanks at the end`
	`< < <`

`Input with absolutely 0 digits in it`
`Well ... apart from that one ...`
	`Input with absolutely < digits in it`
	`Well ... apart from that one ...`

`1 2 4 8 16 32 64 128 256 512 1024`
`2048 4096 8192 16384 32768 65536`
	`< < < > <> << >< <<> <5> 5<< <<<<`
	`<<<> <<>> ><>< <><>< <<>>> >55<>`

As usual:

```
$ ~cs2041/bin/autotest lab06 digits.py
...
$ git add digits.py
$ git commit -a -m "my first python program"
```

Sample solution

```
#!/usr/bin/python

import sys, re

for line in sys.stdin:
    line = re.sub(r'[0-4]', '<', line)
    line = re.sub(r'[6-9]', '>', line)
    sys.stdout.write(line)
# Note above line can also be (Python 3 only):
#   print(line, end='')
```

Sample solution using tr-like approach

```
#!/usr/bin/python

import sys, re

for line in sys.stdin:
    line = re.sub(r'[0-4]', '<', line)
    line = re.sub(r'[6-9]', '>', line)
    sys.stdout.write(line)
# Note above line can also be (Python 3 only):
#   print(line, end='')
```

Challenge Exercise: Repeated Echo in Python

Write a Perl script `echon.py` which given exactly two arguments, an integer *n* and a string, prints the string *n* times. For example:

```
$ ./echon.py 5 hello
hello
hello
hello
hello
hello
```

```
$ ./echon.py 0 nothing
```

```
$ ./echon.py 1 goodbye
goodbye
```

Your script should print an error message if it is not given appropriate arguments. For example:

```
$ ./echon.py
Usage: ./echon.py <number of lines> <string>
```

```
$ ./echon.py 1 2 3
Usage: ./echon.py <number of lines> <string>
```

```
$ ./echon.py Andrew Rocks
./echon.py: argument 1 must be a non-negative integer
```

As usual:

```
$ ~cs2041/bin/autotest lab06 echon.py
...
$ git echon.py
$ git commit -a -m "this is a bad commit message"
```

Sample solution #0 for echon.py **ERROR MISSING FILE: "echonv0.py"** Sample solution #1 for echon.py

```
#!/usr/bin/python

import re, sys

if len(sys.argv) != 3:
    sys.stderr.write("Usage: %s <number of lines> <string>\n" % sys.argv[0])
# Above line can also be in Python3
#   print("Usage: %s <number of lines> <string>" % sys.argv[0], file=sys.stderr)
    sys.exit(1)

if not re.match(r'^\d+$', sys.argv[1]):
    sys.stderr.write("%s: argument 1 must be a non-negative integer\n" % sys.argv[0])
    sys.exit(1)

for i in range(int(sys.argv[1])):
    print(sys.argv[2])
```

Sample solution #2 for echon.py

```
#!/usr/bin/python

import sys

if len(sys.argv) != 3:
    sys.stderr.write("Usage: %s <number of lines> <string>\n" % sys.argv[0])
    sys.exit(1)

# check argument is a positive int  (not required by exercise)
try:
    n = int(sys.argv[1])
except ValueError:
    n = -1
if n < 0:
    sys.stderr.write("%s: argument 1 must be a non-negative integer\n" % sys.argv[0])
    sys.exit(1)

sys.stdout.write((sys.argv[2] + "\n") * n)
```

Challenge Exercise: Tail in Python

Write a Python script `simple_tail.py` to implement the Unix `tail` command. It does not need to implement any command line options. It will be always given one or more files on the command line. It does not have to read from stdin or handle errors.

To assist with testing your solution, there are three small data files: data1.txt (data1.txt), data2.txt (data2.txt), and data3.txt (data3.txt). Copy these file your current directory.

```
$ cp /home/cs2041/public_html/lab/python/simple_tail/t?.txt .
```

```
$ simple_tail.py t1.txt t2.txt t3.txt
Data 1 ... Line 2
Data 1 ... Line 3
Data 1 ... Line 4
Data 1 ... Line 5
Data 1 ... Line 6
Data 1 ... Line 7
Data 1 ... Line 8
Data 1 ... Line 9
Data 1 ... Line 10
Data 1 ... Last line
A one line file.
has
exactly
twelve
lines
with
one
word
on
each
line
```

As usual:

```
$ ~cs2041/bin/autotest lab06 simple_tail.py
...
$ git add simple_tail.py
$ git commit -a -m "my third python program"
```

Sample solution

```
#!/usr/bin/python

import sys,os

# inefficient for large files but simple
for filename in sys.argv[1:]:
    lines = open(filename).readlines()
    tail = lines[-10:]
    for line in tail:
        sys.stdout.write(line)
# Above line can also be in Python3:
#       print(line, end='')
```

More concise solution

```
#!/usr/bin/python

import sys,os

# inefficient for large files but simple
for filename in sys.argv[1:]:
    sys.stdout.write("".join(open(filename).readlines()[-10:]))
```

Challenge Exercise: Whales in Python

Implement Python scripts `whale.py` and `all_whales.py` which count whales as described in the first two lab questions. As usual:

```
$ ~cs2041/bin/autotest lab06 whale.py
...
$ ~cs2041/bin/autotest lab06 all_whales.py
...
$ git add whale.py all_whales.py
$ git commit -a -m "Python & whales!!!"
```

Sample solution for whale.py

```
#!/usr/bin/python

import re, sys

if len(sys.argv) != 2:
    sys.stderr.write("Usage: %s <whale species>\n" % sys.argv[0])
    sys.exit(1)
target_species = sys.argv[1]

n_pods = 0
n_individuals = 0
for line in sys.stdin:
    m = re.search(r'(\d+)\s*(.+)$', line)
    if m:
        count = m.group(1)
        species = m.group(2)
        if species == target_species:
            n_pods += 1
            n_individuals += int(count)
print("%s observations: %d pods, %d individuals" % (target_species, n_pods, n_individuals))
```

Sample solution for all_whales.py

```
#!/usr/bin/python

import re, sys

n_pods = {}
n_individuals = {}
for line in sys.stdin:
    line = line.lower()              # convert line to lower case
    line = line.strip()              # remove surrounding white space
    line = re.sub(r's?$', '', line)  # change to singular
    line = re.sub(r'\s+', ' ', line) # convert sequential white-space charcaters to a single space
    m = re.search(r'(\d+)\s*(.*)$', line)
    if m:
        count = int(m.group(1))
        species = m.group(2)
        if species in n_pods:
            n_pods[species] += 1
            n_individuals[species] += count
        else:
            n_pods[species] = 1
            n_individuals[species] = count
    else:
        print("Sorry couldn't parse: %s" % line)

for species in sorted(n_pods):
    print("%s observations: %d pods, %d individuals" % (species, n_pods[species], n_individuals[species]))
```

Testing

Do your own testing in addition to the autotests.

BTW you can run all the autotests for the lab:

```
$ ~cs2041/bin/autotest lab06
```

Or just one autotest:

```
$ ~cs2041/bin/autotest lab06 whale_1
```

Finalising

You must show your solutions to your tutor and be able to explain how they work. Once your tutor has discussed your answers with you, you should subm them using:

```
$ give cs2041 lab06 whale.pl all_whales.pl prereq.pl [recursive_prereq.pl digits.py echon.py simple_tail.py whale.py all_wha
s.py]
```

Whether you discuss your solutions with your tutor this week or next week, you must submit them before the above deadline.