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

You can Do Anything With Perl

Software Construction (http://www.cse.unsw.edu.au/~cs20

Aims

This week we develop your Perl programming skills.

Assessment

Submission: `give cs2041 lab08 courses.pl lectures0.pl lectures1.pl lectures2.pl [tags.py shortest_path.py]`
Deadline: either during the lab, or Monday October 19 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)

As usual **lab08** sub-directory of your git repo:

```
$ cd
$ cd 2041-labs/lab08
$ gedit courses.pl &
```

Exercise: Scraping Course Codes

Write a Perl script `courses.pl` which prints the course codes with a given prefix of all UNSW courses with lectures on the Kensington Campus this session example:

```
$ ./courses.pl VISN
VISN1101
VISN1111
VISN1221
VISN2111
VISN2211
VISN2231
VISN3111
VISN3211
VISN4003
```

Hints: The course codes with prefix `VISN` can be found in this web page: `http://www.timetable.unsw.edu.au/current/VISNKENS.html` (http://www.timetable.unsw.edu.au/current/VISNKENS.html). You can assume this is the case for all prefixes. You saw how to retrieve a web page using a previous lab. So you can perform this task in less than 10 lines of Perl.

Sample Perl solution

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

foreach $prefix (@ARGV) {
    open my $f, '-|', "wget -q -O- http://www.timetable.unsw.edu.au/current/${prefix}KENS.html" or die;
    while (<$f>) {
        print "$1\n" if />($prefix\d\d\d\d)</;
    }
    close $f;
}
```

Sample Shell solution

```
#!/bin/sh

for course in "$@"
do
    wget -q -O- http://www.timetable.unsw.edu.au/current/${course}KENS.html|
    egrep -oP "$course\d\d\d\d"|
    sort|
    uniq
done
```

As usual:

```
$ ~cs2041/bin/autotest lab08 courses.pl
$ git add courses.pl
$ git commit -a -m "courses.pl was easy - Perl is cool"
```

Exercise: Scraping Lecture Times

Write a Perl script `lectures0.pl` which given course codes as arguments prints details of their lectures.

```
$ ./lectures0.pl COMP2041
COMP2041: S2 Tue 13:00 - 15:00 (Weeks:1-9,10-12), Thu 17:00 - 18:00 (Weeks:1-9,10-12)
$ ./lectures0.pl PSYC1011 COMP9024 COMP3231
PSYC1011: S2 Mon 11:00 - 12:00 (Weeks:1-9,10-12), Wed 14:00 - 15:00 (Weeks:1-9,10-12),
Thu 12:00 - 13:00 (Weeks:1-9,10-12)
PSYC1011: S2 Mon 18:00 - 19:00 (Weeks:1-9,10-12), Wed 18:00 - 19:00 (Weeks:1-9,10-12),
Thu 18:00 - 19:00 (Weeks:1-9,10-12)
COMP9024: S1 Tue 18:00 - 21:00 (Weeks:1-4,5-12)
COMP9024: S2 Thu 18:00 - 21:00 (Weeks:1-9,10-12)
COMP3231: S1 Tue 14:00 - 16:00 (Weeks:1-4,5-12), Wed 16:00 - 17:00 (Weeks:1-4,5-12)
```

Hint: You can assume that a course's lecture times will be found in a web page equivalent to: http://timetable.unsw.edu.au/current/COMP2041.html (http://timetable.unsw.edu.au/current/COMP2041.html). Its difficult to use a regexp to match the line containing the lecture description but you match a previous line, then skip a certain number of lines.

You can also get the teaching period this way. Don't panic if you can't get this quite right your tutor will be generous with hints.

Hint: a hash can be easily used to avoid repeated output.

Hint: make sure you have the URL exactly as above - e.g. don't have repeated slashes (the timetable website uses fragile rewriting rules).

Sample solution for lectures0.pl

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

$base_url = "http://timetable.unsw.edu.au/current";

foreach $course (@ARGV) {
    my %lectures_seen;
    open my $f, '-|', "wget -q -O- $base_url/$course.html" or die "Can not fetch web page for $course: $!";
    while ($line = <$f>) {
        next if $line !~ /href.*>Lecture</;

        my $session = "";
        $line = <$f>;
        $session = $& if $line =~ /[A-Z][0-9]/;

        <$f> foreach 1..4; # skip 4 lines

        $line = <$f>;
        chomp $line;
        $line =~ s/<.*?>//g;
        $line =~ s/^\s*//g;
        $line =~ s/\s*$//g;

        next if !$line;
        next if $lectures_seen{$line}++;
        print "$course: $session $line\n";
    }
    close $f;
}
```

As usual:

```
$ ~cs2041/bin/autotest lab08 lectures0.pl
$ git add lectures0.pl
$ git commit -a -m "lectures0.pl passes autotest"
```

Semi-Challenge Exercise: Lecture Times as Tuples

The output from `lectures0.pl` is (more or less) human readable but is less convenient for other uses. Copy `lectures0.pl` to `lectures1.pl` and modify that if a -d option is specified it prints the hourly details of lectures in the format shown in the examples below:

```
$ ./lectures1.pl COMP2041
COMP2041: S2 Tue 13:00 - 15:00 (Weeks:1-9,10-12), Thu 17:00 - 18:00 (Weeks:1-9,10-12)
$ ./lectures1.pl -d COMP2041
S2 COMP2041 Tue 13
S2 COMP2041 Tue 14
S2 COMP2041 Thu 17
$ ./lectures1.pl -d COMP4121
S2 COMP4121 Mon 11
S2 COMP4121 Mon 12
S2 COMP4121 Thu 9
S2 COMP4121 Thu 10
$ ./lectures1.pl -d COMP1927
X1 COMP1927 Wed 9
X1 COMP1927 Wed 10
X1 COMP1927 Wed 11
X1 COMP1927 Fri 9
X1 COMP1927 Fri 10
X1 COMP1927 Fri 11
S1 COMP1927 Tue 16
S1 COMP1927 Tue 17
S1 COMP1927 Wed 12
S2 COMP1927 Mon 10
S2 COMP1927 Mon 11
S2 COMP1927 Wed 15
S2 COMP1927 Wed 16
$ ./lectures1.pl -d PSYC1011 COMP9024 COMP3231
S2 PSYC1011 Mon 11
S2 PSYC1011 Wed 14
S2 PSYC1011 Thu 12
S2 PSYC1011 Mon 18
S2 PSYC1011 Wed 18
S2 PSYC1011 Thu 18
S1 COMP9024 Tue 18
S1 COMP9024 Tue 19
S1 COMP9024 Tue 20
S2 COMP9024 Thu 18
S2 COMP9024 Thu 19
S2 COMP9024 Thu 20
S1 COMP3231 Tue 14
S1 COMP3231 Tue 15
S1 COMP3231 Wed 16
```

Getting this exercise completely correct is difficult - your tutor will be generous with nearly correct attempts.

See sample solution for lectures2.pl

As usual:

```
$ ~cs2041/bin/autotest lab08 lectures1.pl
$ git add lectures1.pl
$ git commit -a -m "lectures1.pl working for 1 hour lectures"
```

Semi-Challenge Exercise: Lecture Times as a Table

Copy `lectures1.pl` to `lectures2.pl` and modify itso that when a -t option is specified it prints a count of many lectures occur at each day/time in an AS table in the format shown in the example below.

```
$ ./lectures2.pl COMP9020 COMP9021 COMP9024
COMP9020: S1 Fri 14:00 - 17:00 (Weeks:1-4,5-12)
COMP9020: S2 Thu 12:00 - 15:00 (Weeks:1-9,10-12)
COMP9021: S1 Tue 18:00 - 21:00 (Weeks:1-4,5-12)
COMP9021: S2 Thu 18:00 - 21:00 (Weeks:1-9,10-12)
COMP9024: S1 Tue 18:00 - 21:00 (Weeks:1-4,5-12)
COMP9024: S2 Thu 18:00 - 21:00 (Weeks:1-9,10-12)
$ ./lectures2.pl -d COMP9020 COMP9021 COMP9024
S1 COMP9020 Fri 14
S1 COMP9020 Fri 15
S1 COMP9020 Fri 16
S2 COMP9020 Thu 12
S2 COMP9020 Thu 13
S2 COMP9020 Thu 14
S1 COMP9021 Tue 18
S1 COMP9021 Tue 19
S1 COMP9021 Tue 20
S2 COMP9021 Thu 18
S2 COMP9021 Thu 19
S2 COMP9021 Thu 20
S1 COMP9024 Tue 18
S1 COMP9024 Tue 19
S1 COMP9024 Tue 20
S2 COMP9024 Thu 18
S2 COMP9024 Thu 19
S2 COMP9024 Thu 20
$ ./lectures2.pl -t COMP9020 COMP9021 COMP9024
S1       Mon   Tue   Wed   Thu   Fri
09:00
10:00
11:00
12:00
13:00
14:00                             1
15:00                             1
16:00                             1
17:00
18:00           2
19:00           2
20:00           2
S2       Mon   Tue   Wed   Thu   Fri
09:00
10:00
11:00
12:00                       1
13:00                       1
14:00                       1
```

```
15:00
16:00
17:00
18:00                       2
19:00                       2
20:00                       2
$ ./lectures2.pl -t COMP1911 COMP2041 COMP2121 COMP3121 COMP3311 COMP3331 COMP4121 COMP3421 COMP9020 CC
P9021 COMP9024
S1       Mon   Tue   Wed   Thu   Fri
09:00     1
10:00     1
11:00     2           1
12:00     1           1
13:00           1     1     1
14:00           1     1           1
15:00                 1     1     1
16:00                 1           1
17:00                 1
18:00           2
19:00           2
20:00           2
S2       Mon   Tue   Wed   Thu   Fri
09:00                       1
10:00                       1
11:00     1
12:00     1                 1
13:00           1           1
14:00           1     1     2
15:00           1           2
16:00     1     1           1
17:00     1                 1
18:00                       2
19:00                       2
20:00                       2
```

Getting this exercise completely correct is difficult - your tutor will generous with nearly correct attempts.

Sample solution for lectures2.pl

```perl
#!/usr/bin/perl -w
$option_day_time = 0;
$option_table = 0;
$base_url = "http://timetable.unsw.edu.au/current";
$debug = 0;
foreach $arg (@ARGV) {
    if ($arg eq "-d") {
        $option_day_time = 1;
    } elsif ($arg eq "-t") {
        $option_table = 1;
    } elsif ($arg eq "--debug") {
        $debug = 1;
    } else {
        push @courses, $arg;
    }
}
foreach $course (@courses) {
    my %lectures_seen;
    open my $f, '-|', "wget -q -O- $base_url/$course.html" or die "Can not fetch web page for $course: $!";
    while ($line = <$f>) {
        next if $line !~ /href.*>Lecture/;
        print "found line: '$line'\n" if $debug;
        my $session = "";
        $line = <$f>;
        $session = $& if $line =~ /[A-Z][0-9]/;
        <$f> foreach 1..4; # skip 5 lines
        $line = <$f>;
        chomp $line;
        print "raw line = '$line'\n" if $debug;
        $line =~ s/<.*?>//g;
        $line =~ s/^\s*//g;
        $line =~ s/\s*$//g;

        next if !$line;
        next if $lectures_seen{$line}++;

        print "$course: $session $line\n" if !$option_day_time && !$option_table;

        foreach $lecture (split /\), /, $line) {
            print "lecture='$lecture'\n" if $debug;
            my @days = $lecture =~ /\b([a-z]{3})\b/gi ;
            print "\@days='@days'\n" if $debug;

            my ($start_hour,$finish_hour,$finish_minute) = $lecture =~ /(\d\d):\d\d - (\d\d):(\d\d)/ or next;
            $finish_hour++ if $finish_minute ne "00";
            print "start_hour=$start_hour finish_hour=$finish_hour\n" if $debug;

            foreach $day (@days) {
                foreach $time ($start_hour..$finish_hour-1) {
                    if (!$lecture{$session}{$day}{$time}{$course}++ and $option_day_time) {
                        print "$session $course $day $time\n";
                    }
                }
            }
        }
    }
    close $f;
}

if ($option_table) {
    foreach $session (sort keys %lecture) {
        my @days =qw/Mon Tue Wed Thu Fri/;
        my $width = 6;
        printf "%-${width}s", $session;
        printf "%${width}s",$_ foreach @days;
        print "\n";
        foreach $time (9..20) {
            printf "%02d:00", $time;
            foreach $day (@days) {
                my $n_lectures =  keys %{$lecture{$session}{$day}{$time}};
                printf "%${width}s", $n_lectures || "";
            }
            print "\n";
        }
    }
}
```

As usual:

```
$ ~cs2041/bin/autotest lab08 lectures2.pl
$ git add lectures2.pl
$ git commit -a -m "lectures2.pl half working"
```

Challenge Exercise: Extracting Tags in Python

The introduction to Python in lectures will come later.

The example Python scripts (/~cs2041/lec/python/examples.index.html) and links to external Python resources should help - but you may need more info Google is your friend. Write a Python program, `tags.py` which given the URL of a web page fetches it by running *wget* and prints the HTML tags it uses.

Don't count closing tags (e.g. `</a>`)

The tag should be converted to lower case and printed in sorted order with a count of often each is used.

You are expected (not required) to use regexes but there are problems processing HTML this way - don't be surprised if its hard to get right.

Hint: make sure you don't print tags within HTML comments.

For example:

```
$ ./tags.py http://www.cse.unsw.edu.au/~cs2041/intro.html
a 20
b 8
body 1
br 17
center 1
div 2
em 2
font 15
h1 1
h3 18
head 1
hr 2
html 1
li 52
link 1
ol 2
p 32
pre 1
script 1
small 15
table 3
tbody 1
td 20
th 5
title 1
tr 9
ul 11
$ ./tags.py http://www.cse.unsw.edu.au/~cs2041/15s2/
a 138
b 3
body 1
br 27
code 1
div 9
em 36
head 1
html 1
link 1
p 1
script 1
small 3
table 3
tbody 3
td 162
th 11
```

```
thead 3
title 1
tr 29
$ ./tags.py http://www.cse.unsw.edu.au
a 151
body 1
br 2
div 427
fieldset 1
form 1
h2 27
h3 1
head 1
html 1
iframe 1
img 24
input 6
label 2
legend 1
li 81
link 9
meta 11
noscript 1
p 15
script 10
span 38
title 1
ul 13
```

Add an `-f` option to `tags.py` which indicates the tags are to printed in order of frequency. For example:

```
$ ./tags.py -f http://www.cse.unsw.edu.au/~cs2041/intro.html
pre 1
h1 1
title 1
tbody 1
html 1
body 1
head 1
link 1
center 1
script 1
em 2
hr 2
ol 2
div 2
table 3
th 5
b 8
tr 9
ul 11
font 15
small 15
br 17
h3 18
td 20
a 20
p 32
li 52
```

Sample solution using wget

```
#!/usr/bin/python
# written by andrewt@cse.unsw.edu.au as a COMP2041 example
# fetch specified web pages and count the HTML tags in them
#
# There are python libraries which provide a  better way to fetch web pages
#
# subprocess.check_output was introduced in Python 2.7.
# In previous version you might use:
# webpage = subprocess.Popen(["wget","-q","-O-",url], stdout=subprocess.PIPE).communicate()[0]
#
#
# The regex code below doesn't handle a number of cases.  It is often
# better to use a library to properly parse HTML before processing it.
# But beware illegal HTML is common & often causes problems for parsers.

import sys, re, subprocess, collections
tag_count = collections.defaultdict(lambda:0)

for url in sys.argv[1:]:
    webpage = subprocess.check_output(["wget","-q","-O-",url], universal_newlines=True)
    webpage = re.sub(r'<!--.*?-->','', webpage) # remove comments
    webpage = webpage.lower()
    for tag in re.findall(r'<\s*(\w+)', webpage):
        if tag:
            tag_count[tag] += 1
for tag in sorted(tag_count.keys()):
    print("%s %d"%(tag, tag_count[tag]))
```

Sample solution using urllib & adding -f option

```
#!/usr/bin/python

# writen by andrewt@cse.unsw.edu.au as a COMP2041 example
# fetch a web page remove HTML tags, constants,
# text between script blank lines
# and print non-empty lines

# The regex code below doesn't handle a number of cases.  It is often
# better to use a library to properly parse HTML before processing it.
# But beware illegal HTML is common & often causes problems for parsers.

import sys, re, collections

# urllib package names changed in Python 3
try:
    from urllib import urlopen # Python 2
except ImportError:
    from urllib.request import urlopen # Python 3

if len(sys.argv) > 1 and sys.argv[1] == "-f":
    sort_by_frequency = 1
    urls = sys.argv[2:]
else:
    sort_by_frequency = 0
    urls = sys.argv[1:]

tag_count = collections.defaultdict(lambda:0)
for url in urls:
    webpage = urlopen(url).read().decode('utf-8')  # would be better to check encoding rather than assume utf-8
    webpage = re.sub(r'<!--.*?-->','', webpage)    # remove comments
    webpage = webpage.lower()
    for tag in re.findall(r'<\s*(\w+)', webpage):
        if tag:
            tag_count[tag] += 1

if sort_by_frequency:
    ordered_tags = sorted(list(tag_count.keys()), key=lambda t: tag_count[t])
else:
    ordered_tags = sorted(tag_count.keys())
for tag in ordered_tags:
    print("%s %d"%(tag, tag_count[tag]))
```

Sample solution using Beautiful soup to parse HTML

```
#!/usr/bin/python2
# writen by andrewt@cse.unsw.edu.au as a COMP2041 example
# fetch a web page remove HTML tags, constants,
# text between script blank lines
# and print non-empty lines

import sys, re, collections

from urllib import urlopen
import BeautifulSoup

# on Python 3 instead do
# from urllib.request import urlopen # Python 3
# import bs4 as BeautifulSoup
# and change BeautifulSoup(webpage) to BeautifulSoup(webpage, "lxml")

if len(sys.argv) > 1 and sys.argv[1] == "-f":
    sort_by_frequency = 1
    urls = sys.argv[2:]
else:
    sort_by_frequency = 0
    urls = sys.argv[1:]

tag_count = collections.defaultdict(lambda:0)
for url in urls:
    webpage = urlopen(url).read()
    soup = BeautifulSoup.BeautifulSoup(webpage)
    for tag in soup.findAll():
        tag_count[tag.name] += 1

if sort_by_frequency:
    ordered_tags = sorted(tag_count.keys(), key=lambda t: tag_count[t])
else:
    ordered_tags = sorted(tag_count.keys())
for tag in ordered_tags:
    print("%s %d"%(tag, tag_count[tag]))
```

Hint: see last week's tute for a sample solution in Perl. As usual:

```
$ ~cs2041/bin/autotest lab08 tags.py
$ git add tags.py
$ git commit -a -m "tags.py almost working"
```

Challenge Exercise: Finding the Shortest Journey in Python

Write a Python program shortest_path.py that given the road distances between a number of towns (on standard input) calculates the shortest journey be two towns specified as arguments. Here is an example of how your program should behave.

```
$ ./shortest_path.py Parkes Gilgandra
 Bourke Broken-Hill   217
Bourke Dubbo         23
Bourke Gilgandra      62
Bourke Parkes        71
Canowindra Dubbo      35
Canowindra Gilgandra  13
Canowindra Parkes    112
Dubbo Gilgandra       91
Dubbo Parkes         57
<cntrl-d>
Shortest route is length = 105: Parkes Dubbo Canowindra Gilgandra.
```

Hints: Python's strings have a split method (http://en.wikibooks.org/wiki/Python_Programming/Strings#split.2C_splitlines) which can break up input lines easy to implement Sets in python (http://en.wikibooks.org/wiki/Python_Programming/Sets).

Python sample solution translated from Perl

```
#!/usr/bin/python

import fileinput, re, sys, collections

(start,finish) = sys.argv[1:]

distance = {}
for line in sys.stdin:
    (town1,town2,dist) = line.split()
    distance[(town1,town2)] = int(dist)
    distance[(town2,town1)] = int(dist)

shortest_journey = {start:0}
route = {start:''}
unprocessed_towns = set(d[0] for d in distance.keys())
next_town = start
while next_town and next_town != finish:
    unprocessed_towns.remove(next_town)
    for town in unprocessed_towns:
        if (next_town,town) in distance:
            d = shortest_journey[next_town] + distance[(next_town,town)]
            if town not in shortest_journey or shortest_journey[town] > d:
                shortest_journey[town] = d
                route[town] = route[next_town] + " " + next_town
    min_distance = 1e99  # must be larger than any possible distance
    next_town = ""
    for town in unprocessed_towns:
        if town in shortest_journey and shortest_journey[town] < min_distance:
            min_distance = shortest_journey[town]
            next_town = town

if finish not in shortest_journey:
    print("No route from %s to %s" % (start, finish))
else:
    print("Shortest route is length = %s:%s %s." % (shortest_journey[finish], route[finish], finish))
```

More python-ish sample solution

```
#!/usr/bin/python

import fileinput, re, sys, collections

(start,finish) = sys.argv[1:]
distance = {}
for line in sys.stdin:
    (town1,town2,dist) = line.split()
    distance[(town1,town2)] = int(dist)
    distance[(town2,town1)] = int(dist)

shortest_journey = {start:0}
route = {start:''}
unprocessed_towns = set(d[0] for d in distance.keys())
next_town = start
while next_town and next_town != finish:
    unprocessed_towns.remove(next_town)
    for town in unprocessed_towns:
        if (next_town,town) in distance:
            d = shortest_journey[next_town] + distance[(next_town,town)]
            if town not in shortest_journey or shortest_journey[town] > d:
                shortest_journey[town] = d
                route[town] = route[next_town] + " " + next_town
    next_town = min((shortest_journey[town],town) for town in unprocessed_towns if town in shortest_journey)[1]

if finish not in shortest_journey:
    print("No route from %s to %s" % (start, finish))
else:
    print("Shortest route is length = %s: %s %s." % (shortest_journey[finish], route[finish], finish))
```

Slightly different python-ish sample solution

```
#!/usr/bin/python

import fileinput, re, sys, collections

(start,finish) = sys.argv[1:]

distance = {}
for line in sys.stdin:
    (town1,town2,dist) = line.split()
    distance[(town1,town2)] = int(dist)
    distance[(town2,town1)] = int(dist)

shortest_journey = {start:0}
route = {start:''}
unprocessed_towns = set(d[0] for d in distance.keys())
next_town = start
while next_town and next_town != finish:
    unprocessed_towns.remove(next_town)
    for town in unprocessed_towns:
        if (next_town,town) in distance:
            d = shortest_journey[next_town] + distance[(next_town,town)]
            if town not in shortest_journey or shortest_journey[town] > d:
                shortest_journey[town] = d
                route[town] = route[next_town] + " " + next_town
    next_town = min(unprocessed_towns&set(shortest_journey.keys()), key=shortest_journey.get)

if finish not in shortest_journey:
    print("No route from %s to %s" % (start, finish))
else:
    print("Shortest route is length = %s: %s %s." % (shortest_journey[finish], route[finish], finish))
```

Hint: see last week's tute for a sample solution in Perl. As usual:

```
$ ~cs2041/bin/autotest lab08 shortest_path.py
$ git add shortest_path.py
$ git commit -a -m "my shortest_path.py rocks!"
```

Testing

You will need to do your own testing but to assist you as usual some autotest tests are available for this lab.

To run all tests:

```
$ ~cs2041/bin/autotest lab08
```

You can run a single test if you also pass the test label as the second argument to autotest. For example, to run just test `courses_2` type:

```
$ ~cs2041/bin/autotest lab08 courses_2
```

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 lab08 courses.pl lectures0.pl lectures1.pl lectures2.pl [tags.py shortest_path.py]
```

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