# Week 06 Laboratory

## Sample Solutions

## Objectives

- Getting used to more Perl features.

## Preparation

Before the lab you should re-read the relevant lecture slides and their accompanying examples.

## Getting Started

Create a new directory for this lab called `lab06` by typing:

```
$ mkdir lab06
```

Change to this directory by typing:

```
$ cd lab06
```

## Exercise: How many Orcas in that File?

A citizen science project monitoring whale populations has files of containing large numbers of whale observations. Each line in these files contains:

- the date the observation was made
- the number of whales in the pod ("pod" is the collective number for a group of whales)
- the species of whale

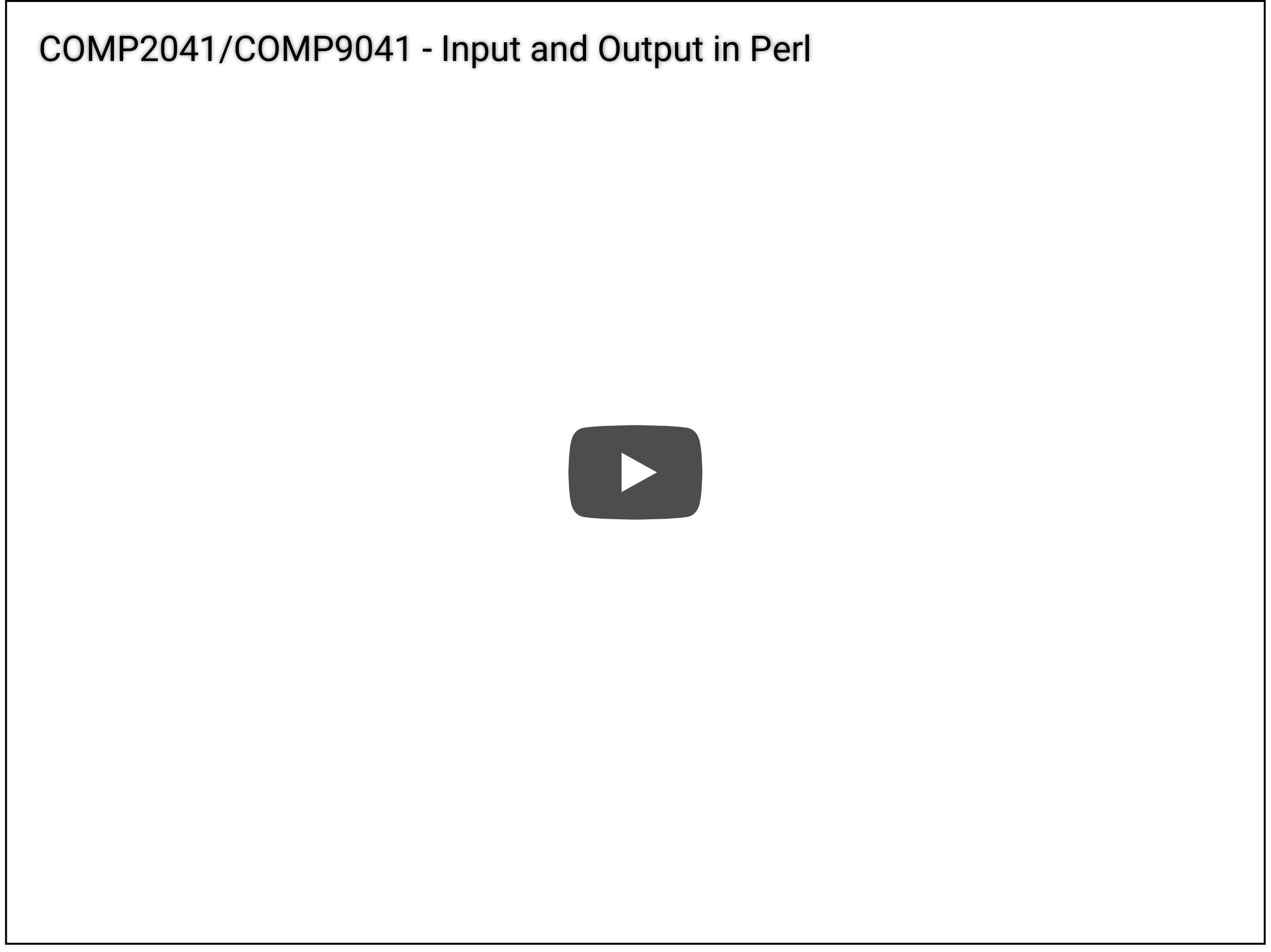
Here is a file of example data which you can download to test your program. It looks like this:

```
$ head whales.txt
18/01/18  9 Pygmy right whale
01/09/17 21 Southern right whale
16/02/18  4 Striped dolphin
02/07/17  4 Common dolphin
19/02/18  4 Blue whale
21/02/18 38 Dwarf sperm whale
14/06/17 29 Southern right whale
20/06/17  3 Spinner dolphin
22/07/17 34 Indo-Pacific humpbacked dolphin
20/03/18  7 Long-finned pilot whale
```

Write a Perl program **orca.pl** which prints a count of the number of observations of Orcas in the file.

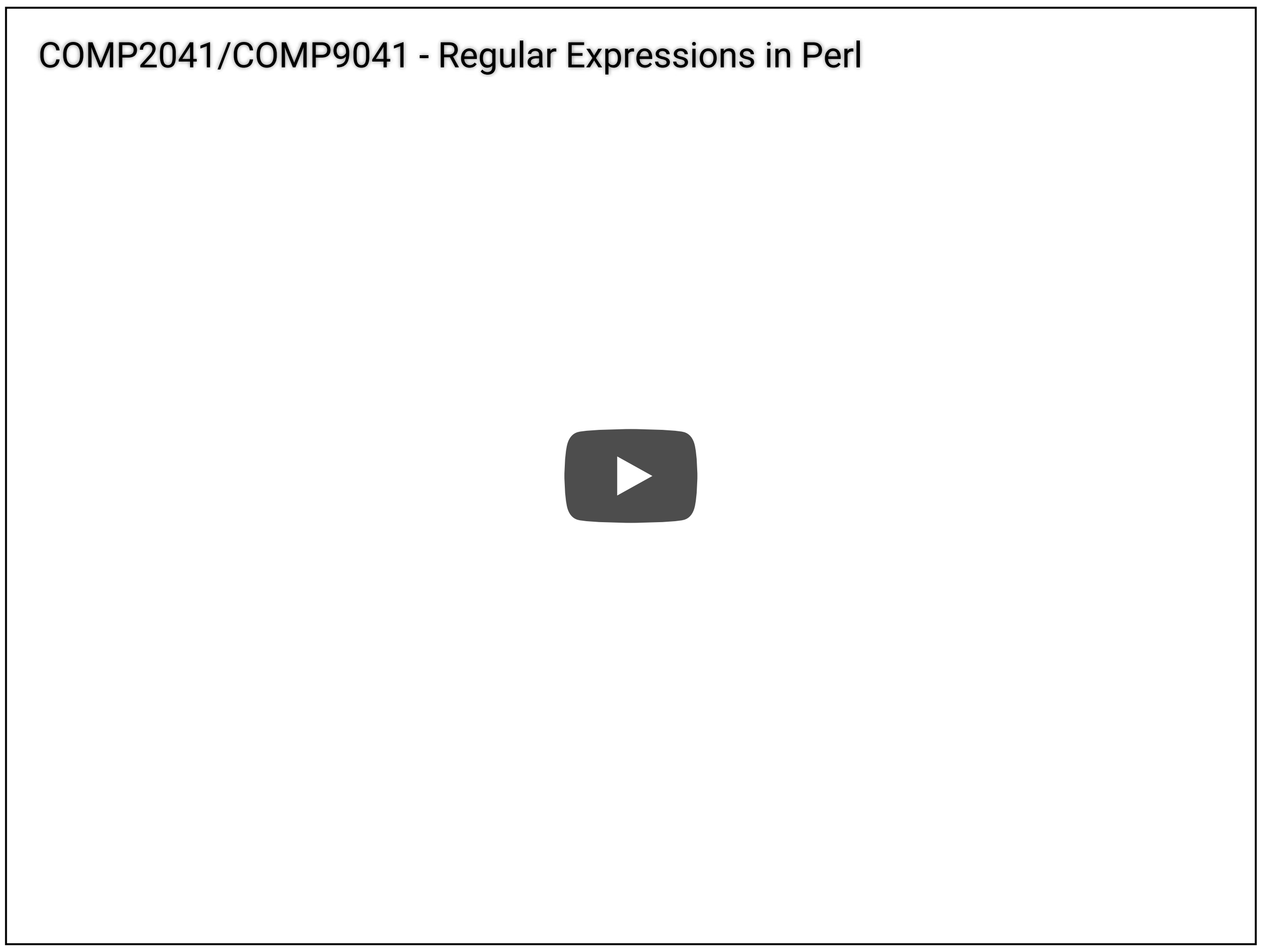
```
$ ./orca.pl whales.txt
1245 Orcas reported in whales.txt
```

Note each line records one pod of whales Your program should print the number of individual Orca observed - it needs to sum the number of individual whales in each Orca pod. **Hint**: if you need help with Perl I/0 COMP2041 tutor Sabrina Rispin takes you through the basics in this video:



And Jashank Jeremy takes you through the basics of Perl strings in this video:

And regexes in this video:



When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest orca
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab06_orca orca.pl
```

Sample solution for `orca.pl`

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

# Count Orca observations in 1 or more files
# andrewt@unsw.edu.au for COMP[29]041 lab exercise

for $filename (@ARGV) {
    my $n_orcas = 0;
    my $line;

    open my $f, $filename or die "Can not open $filename\n";

    while ($line = <$f>) {
        if ($line =~ /(\d+)\s+Orca\s*$/i) {
            $n_orcas += $1;
        }
    }

    close $f;

    print "$n_orcas Orcas reported in $filename\n";
}
```

Alternative solution for `orca.pl`

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

# Count Orca observations in 1 or more files
# andrewt@unsw.edu.au for COMP[29]041 lab exercise
# Terse less-readble solution using $_ and reversed for loop

for $filename (@ARGV) {
    my $n_orcas = 0;
    open my $f, $filename or die "Can not open $filename\n";
    /\s(\d+)\s+Orca\s*$/i and $n_orcas += $1 for <$f>;
    close $f;
    print "$n_orcas Orcas reported in $filename\n";
}
```

## Exercise: Counting A Whale Species

Write a Perl program **species_count.pl** which take as its first argument a whale species and its second argument a filename. Your program should print the number of pods of that species reported in the file and count of the total individuals of that species.

The file will be in the same format as the last exercise. For example:

```
$ ./species_count.pl Orca whales.txt
Orca observations: 53 pods, 1245 individuals
$ ./species_count.pl "Indo-Pacific humpbacked dolphin" whales.txt
Indo-Pacific humpbacked dolphin observations: 43 pods, 897 individual
s
```

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest species_count
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab06_species_count species_count.pl
```

Sample solution for `species_count.pl`

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

# Count whale observations in 1 or more files
# andrewt@unsw.edu.au for COMP[29]041 lab exercise

die "Usage: $0 <whale species> <files>\n" if !@ARGV;

$target_species = shift @ARGV;

for $filename (@ARGV) {
    my $n_pods = 0;
    my $n_individuals = 0;

    open my $f, $filename or die "Can not open $filename\n";

    while ($line = <$f>) {
        if ($line =~ /\s+(\d+)\s+(.+)\s*$/i) {
            my $count = $1;
            my $species = $2;
            if ($species eq $target_species) {
                $n_pods++;
                $n_individuals += $count;
            }
        } else {
            warn "Couldn't parse: $line\n";
        }
    }

    close $f;

    print "$target_species observations: $n_pods pods, $n_individuals individuals\n";
}
```

## Exercise: Printing A summary of All Whale Species

Write a Perl program **whale_summary.pl** which take as its first argument a filename.
**whale_summary** should **print** for every whale species in the file how many pods were seen and how many total whales were seen of that species.

Whale species should be listed in alphabetic 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.

For example:

```
$ ./whale_summary.pl whales.txt
blue whale observations: 51 pods, 1171 individuals
bryde's whale observations: 47 pods, 1023 individuals
coastal bottlenose dolphin observations: 53 pods, 1169 individuals
common dolphin observations: 62 pods, 1232 individuals
dwarf minke whale observations: 49 pods, 1080 individuals
dwarf sperm whale observations: 58 pods, 1272 individuals
fin whale observations: 65 pods, 1251 individuals
humpback whale observations: 55 pods, 1071 individuals
indo-pacific humpbacked dolphin observations: 43 pods, 897 individual
s
long-finned pilot whale observations: 50 pods, 996 individuals
orca observations: 53 pods, 1245 individuals
pygmy right whale observations: 59 pods, 1260 individuals
pygmy sperm whale observations: 57 pods, 1346 individuals
sei whale observations: 55 pods, 929 individuals
short-finned pilot whale observations: 68 pods, 1344 individuals
southern right whale observations: 55 pods, 1252 individuals
spinner dolphin observations: 52 pods, 1118 individuals
striped dolphin observations: 68 pods, 1337 individuals
$ cat messy_whales.txt
19/02/18  4 blue WHALE
11/02/18  6 dwarf   minke    whale
26/04/18 19    orcas
19/12/17 37 Dwarf minke whales
02/09/17 25 Dwarf   minke   whale
```

**Hint**: if you need help with Perl hashes COMP2041 tutor Sabrina Rispin takes you through the basics in this video:

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest whale_summary
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab06_whale_summary whale_summary.pl
```

Sample solution for `whale_summary.pl`

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

# Print a summary of whale observations in 1 or more files
# andrewt@unsw.edu.au for COMP[29]041 lab exercise

for $filename (@ARGV) {
    open my $f, $filename or die "Can not open $filename\n";

    while ($line = <$f>) {
        $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 ($line =~ /\s+(\d+)\s+(.+)\s*$/) {
            my $count = $1;
            my $species = $2;

            $species = lc $species;     # convert line to lower case
            $species =~ s/\s+$//;    # remove trailing white space
            $species =~ s/s?$//;     # change to singular
            $species =~ s/\s+/ /g;   # convert sequential white-space characters to a single space

            $n_pods{$species}++;
            $n_individuals{$species} += $count;
        } else {
            warn "Couldn't parse: $line\n";
        }
    }

    close $f;

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

## Exercise: Course 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
COMP1511
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.

Note the 2019 handbook is availabl don't use it - use the 2018 handbook. **Hints** The UNSW handbook uses separate web pages for undergraduate and postgraduate handbooks and you may need to extract prerequisites from either or both.

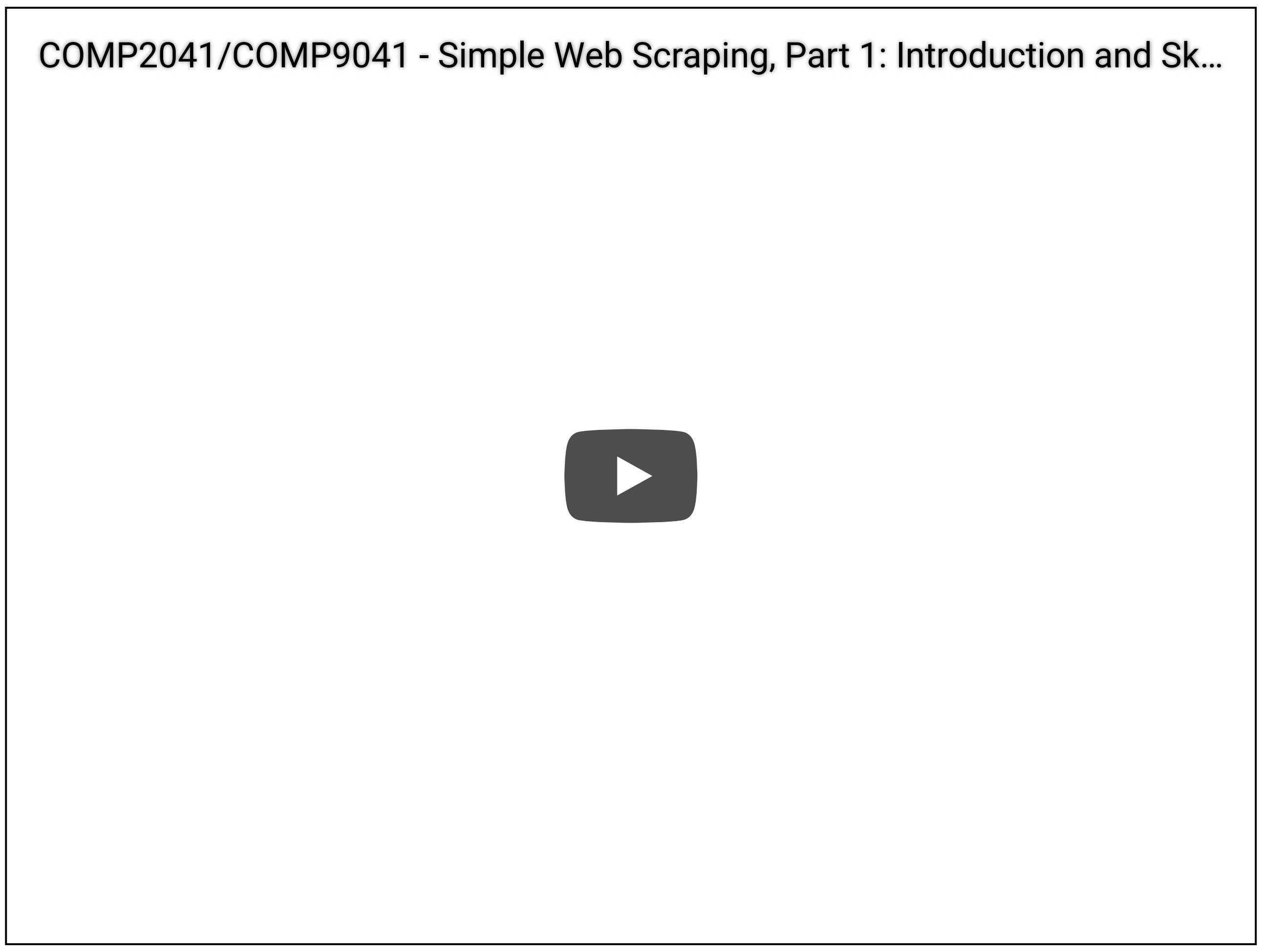
A simple way (but not the best way) to access a web page from Perl is like this:

```
$url = "http://www.handbook.unsw.edu.au/postgraduate/courses/2018/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's easy in Perl to skip lines until you find one specifying prerequisites.

It's easy in Perl to remove part of a line.

It's easy in Perl to remove HTML tags. **Hint**: if you need help these three videos by Emily Chen take through a similar task:


COMP2041/COMP9041 - Simple Web Scraping, Part 1: Introduction and Sk...

COMP2041/COMP9041 - Simple Web Scraping, Part 2: Diving into HTML

COMP2041/COMP9041 - Simple Web Scraping, Part 3: Extending with Opti...

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest prereq
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab06_prereq prereq.pl
```

Sample solution for `prereq.pl`

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

$year = 2018;
$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";
}
```

## Challenge Exercise: Recursive Course Prerequisites

Add a -r (for recursive ) flag to `prereq.pl` so it also lists all courses which can be used to meet a prerequisites requirement of any course that can be used to meet prerequisite requirements and so on.
Call your new program `recursive_prereq.pl`.

For example:

```
$ ./recursive_prereq.pl -r COMP9243
COMP1511
COMP1521
COMP1911
COMP1917
COMP1921
COMP1927
COMP2121
COMP2521
COMP3231
COMP3331
COMP3891
COMP9021
COMP9024
COMP9032
COMP9201
COMP9283
COMP9331
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 need to find their prerequisites (don't look up old handbooks).

Beware infinite loops!

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest recursive_prereq
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab06_recursive_prereq recursive_prereq.pl
```

Sample solution for `recursive_prereq.pl`

```perl
#!/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;
```

## Submission

When you are finished each exercises make sure you submit your work by running **give**.
You can run **give** multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Sunday 2 September 23:59:59** to submit your work.

You cannot obtain marks by e-mailing lab work to tutors or lecturers.

You check the files you have submitted here

Automarking will be run by the lecturer several days after the submission deadline for the test, using test cases that you haven't seen: different to the test cases `autotest` runs for you.

(Hint: do your own testing as well as running `autotest`)

After automarking is run by the lecturer you can view it here the resulting mark will also be available via via give's web interface

### Lab Marks

When all components of a lab are automarked you should be able to view the the marks via give's web interface or by running this command on a CSE machine:

```
$ 2041 classrun -sturec
```

The lab exercises for each week are worth in total 1 mark.

The best 10 of your 11 lab marks will be summed to give you a mark out of 9. If their sum exceeds 9 - your mark will be capped at 9.

- You can obtain full marks for the labs without completing challenge exercises
- You can miss 1 lab without affecting your mark.