COMP[29]041 16s2 (http://www.cse.unsw.edu.au/~cs2041/16s2/) **Filters and Regexps** Software Construction (http://www.cse.unsw.edu.au/~cs2(

Aims

This exercise aims to give you practice with using the Unix shell for processing collections of files, and in using Unix filters for performing analyses of files.

Assessment

Submission: `give cs2041 lab02 lab02.txt`
Deadline: either during the lab, or Monday 8 August 11:59pm (midnight)
Assessment: Make sure that you are familiar with the lab assessment criteria (lab/assessment.html).

Tasks

These week's questions mostly require you to find a shell command or pipeline to solve the task.
As you go through the lab, after you determine the answer copy the pipeline and its output into a file named `lab02.txt`. Use this template (lab/lab02.txt that this lab page contains hints that the template does not, so keep this page open.

After your tutor has checked your work, submit this file using the command `give cs2041 lab02 lab02.txt`.

Searching a Dictionary with `Less`

On most Unix systems you will find one or more dictionaries containing many thousands of words usually in the directories. `/usr/share/dict/` or `/usr/dict/`.
We copied one such dictionary (lab/sh/dictionary/words.txt) for this lab exercise. To save repeatedly typing its full pathname for these exercises, ma symbolic link to the file with this command:

```
$ ln -s ~cs2041/public_html/lab/sh/dictionary/words.txt
```

(this creates a link to the file in the current directory pointing to the original file, without taking up the space of a complete copy)

The preferred viewer for (potentially very long) text files is `less`, whose name is a lame pun on that of its predecessor, `more`. `less` has regular expression search capabilities.

Within `less`:

- the space bar advances a page at a time
- **b** goes back a page
- a *number* followed by `G` goes to that line (e.g. `50G` goes to line 50)
- `G` by itself goes to the end of the file
- slash (/) followed by a *regexp* finds the next instance of the *regexp* in the file (there is no terminating slash)
- question-mark (?) followed by a *regexp* finds the previous instance of the *regexp* in the file (no terminating '?')

For regular-expression searching, make sure that you understand where `less` leaves the matching line on the screen.

Access the dictionary via the command

```
$ less words.txt
```

a. How do you use `less` to find word is on line 2000 (it's at the top of the page)?

```
Less keystrokes: 2000G
Word: Algonquian
```

b. How do you use `less` to find the first word in the dictionary containing a 'z' `z`

```
Less keystrokes: 1G then /z
Word: Abkhaz
```

c. How do you use `less` to find the last word in the dictionary that contains the substring `ooz` ?

```
Less keystrokes: G then ?ooz
Word: zoozoos
```

Applying Unix Tools to a Dictionary

Write shell pipelines using `egrep`, `cat` and `wc` which answer the following questions.

Provide both the command/pipeline and its output.

a. How many total words does the file `words.txt` contain?

```
$ wc -w words.txt
390582 words.txt
```

b. Note that some of these words are derivates of other words (e.g. "Aberdeen" and "Aberdeen's"). Maybe these shouldn't be included in the wor count. How many total words, excluding those ending in "'s", are there in the dictionary?

```
$ egrep -v "'s$" words.txt | wc -w
299845
```

c. How could you use `cat -n` and `egrep` to find out which word is on line 100000? (you'll need to check the exact output format of `cat -n`)

```
$ cat -n words.txt | egrep '^100000'
100000  adviser
```

Not that each line coming out of the `cat` starts with some spaces, followed by a number, followed by a TAB character. You can explicitly ma the TAB if you want, but you'll need to type control-V followed by TAB in order to get a TAB to appear in the command line.n line 100000

d. How could you use `cat -n` and `egrep` to print the 700th line, and no other lines? (you'll need to check the exact output format of `cat -n`)

```
$ cat -n words.txt | egrep '^ *700[^0-9]'
   700  Adirondack
```

Note: \s may not work in CSE labs. cat -n words.txt | egrep '[[:space:]]700[[:space:]]' is another solution to replace \s.

e. How do you use `head` and `tail` to find out what word is on line 200000

```
$ head -200000 words.txt | tail -1
geodynamics's
```

Finding Unusual Words With Regular Expressions

Now consider how we might locate various "unusual" words in the dictionary using `egrep`:

a. which words contain the characters "lmn" consecutively?

```
$ egrep 'lmn' words.txt
Selmner
Selmner's
almner
almners
calmness
calmness's
calmnesses
```

b. how many words contain "zz", but do not end in apostrophe-s ('s)?

```
$ egrep 'zz' words.txt | egrep -v "'s$"|wc -w
628
```

or

```
$ egrep 'zz' words.txt | egrep -c -v "'s$"
628
```

c. how many words contain four consecutive vowels?

```
$ egrep '[aeiou][aeiou][aeiou][aeiou]' words.txt|wc
    205     205    2296
```

or

```
$ egrep -c '[aeiou]{4}' words.txt
205
```

or ignoring case:

```
$ egrep '[aeiouAEIOU][aeiouAEIOU][aeiouAEIOU][aeiouAEIOU]' words.txt|wc
    207     207    2308
```

or

```
$ egrep -ic '[aeiou]{4}' words.txt
207
```

d. which English words contain all 5 english vowels "aeiou" in that order?
Note, the word may contain more than 5 vowels but it must contain aeiou in that order. e.g. ub**a**isda**e**ucx**io**fgde**u**sc

```
$ egrep -i 'a.*e.*i.*o.*u' words.txt
abstemious
abstemiously
abstemiousness
abstemiousness's
abstemiousnesses
abstentious
adenocarcinomatous
adventitious
adventitiously
adventitiousness
adventitiousness's
adventitiousnesses
aeruginous
amentiferous
androdioecious
andromonoecious
anemophilous
antenniferous
antireligious
arenicolous
argentiferous
arsenious
arteriovenous
```

```
asclepiadaceous
autoecious
autoeciously
bacteriophagous
caesalpiniaceous
caesious
cavernicolous
chaetiferous
facetious
...
```

e. how many English words contain all 5 english vowels "aeiou" in that order?

```
$ egrep -i 'a.*e.*i.*o.*u' words.txt|wc -w
53
```

f. **Challenge** which English words contain exactly 5 vowels and the vowels are "aeiou" in that order?

```
$ egrep -i '^[^aeiou]*a[^aeiou]*e[^aeiou]*i[^aeiou]*o[^aeiou]*u[^aeiou]*$' words.txt
abstemious
abstemiously
abstentious
arsenious
caesious
facetious
facetiously
```

g. Still on words, we want to know how many words have another 9-letter dictionary word embedded in them.
Place all the 9-letter words in the dictionary in a file called `words9.txt` . How to do this? Use a egrep pattern anchored at both ends, that matc
letters.

Don't include words which contain upper case characters.

Don't include words which contain any non-letter in them, We don't want to clude words such as `Bernouilli's`).

Redirect the output of the command using the > shell notation. How many of these words are there?

Now we want to use the list of words as a (huge) search pattern. `fgrep` uses a highly optimised algorithm to do this. Its usage is

```
fgrep –f WordsToFind FilesToSearch...
```

e.g.

```
fgrep –f words9.txt words.txt
```

Devise a pipeline that uses your word list (`words9.txt`) and `words.txt` to count the number of words that have a 9-letter word as a substring
example "luminesce" is a substring of "photoluminescence".

Again we don't want to match words that contain upper-case letters or non-letters.

In other words how many 10+ letter lower case words which contain a 9 character lower-case word.

Note there are a small number of non-English words in the dictionary which have diacritical marks (http://en.wikipedia.org/wiki/Diacritic) whic
encoded as a separate character in ASCII - your regular expressions don't need to handle these correctly.

```
# Create the words9.txt file
# Note that this ignores words ending in 's
egrep '^[a-z]{9}$' words.txt > words9.txt
# Get the words containg these as a substring
# First stage: extract all words that contain a "words9.txt" word as a
#              substring or are equal to a "words9.txt" word
# Second stage: eliminate all nine-character words (i.e. drop all
#              of the extracted words that come from the "words9.txt" file)
fgrep -f words9.txt words.txt | egrep -iv '^[a-z]{9}$'
```

```
$ fgrep -f words9.txt words.txt |egrep -v '[^a-z]'|egrep '[a-z]{10}'|wc -w
30117
```

egrep ing MPs

The file `/home/cs2041/public_html/lab/sh/parliament/parliament.txt` (lab/sh/parliament/parliament.txt) contains a list of the members of the Australian House of Representatives (MPs). For example:

```
$ head parliament.txt
Hon Tony Abbott: Member for Warringah, New South Wales
Hon Anthony Albanese: Member for Grayndler, New South Wales
Mr John Alexander OAM: Member for Bennelong, New South Wales
Hon Karen Andrews: Member for McPherson, Queensland
Hon Kevin Andrews: Member for Menzies, Victoria
Mr Adam Bandt: Member for Melbourne, Victoria
Ms Julia Banks: Member for Chisholm, Victoria
Hon Sharon Bird: Member for Cunningham, New South Wales
Hon Julie Bishop: Member for Curtin, Western Australia
Hon Chris Bowen: Member for McMahon, New South Wales
```

Each line contains the name of the MP followed by the electorate they represent.

a. Write an egrep command that will print all the lines in the file where the electorate begins with W. Hint the output should be:

```
Hon Tony Abbott: Member for Warringah, New South Wales
Mr Scott Buchholz: Member for Wright, Queensland
Hon Tony Burke: Member for Watson, New South Wales
Mr Nick Champion: Member for Wakefield, South Australia
Mr Peter Khalil: Member for Wills, Victoria
Mr Llew O'Brien: Member for Wide Bay, Queensland
Ms Anne Stanley: Member for Werriwa, New South Wales
Hon Dan Tehan: Member for Wannon, Victoria
Hon Malcolm Turnbull: Member for Wentworth, New South Wales
```

```
$ egrep 'Member for W' parliament.txt
Hon Tony Abbott: Member for Warringah, New South Wales
Mr Scott Buchholz: Member for Wright, Queensland
Hon Tony Burke: Member for Watson, New South Wales
Mr Nick Champion: Member for Wakefield, South Australia
Mr Peter Khalil: Member for Wills, Victoria
Mr Llew O'Brien: Member for Wide Bay, Queensland
Ms Anne Stanley: Member for Werriwa, New South Wales
Hon Dan Tehan: Member for Wannon, Victoria
Hon Malcolm Turnbull: Member for Wentworth, New South Wales
```

b. Write an egrep command that will list all the lines in the file where the MP's first name is Andrew. Hint the output should be:

```
Mr Andrew Broad: Member for Mallee, Victoria
Mr Andrew Gee: Member for Calare, New South Wales
Mr Andrew Giles: Member for Scullin, Victoria
Mr Andrew Hastie: Member for Canning, Western Australia
Mr Andrew Laming: Member for Bowman, Queensland
Hon Dr Andrew Leigh: Member for Fenner, Australian Capital Territory
Mr Andrew Wallace: Member for Fisher, Queensland
Mr Andrew Wilkie: Member for Denison, Tasmania
```

```
$ egrep ' Andrew ' parliament.txt
Mr Andrew Broad: Member for Mallee, Victoria
Mr Andrew Gee: Member for Calare, New South Wales
Mr Andrew Giles: Member for Scullin, Victoria
Mr Andrew Hastie: Member for Canning, Western Australia
Mr Andrew Laming: Member for Bowman, Queensland
Hon Dr Andrew Leigh: Member for Fenner, Australian Capital Territory
Mr Andrew Wallace: Member for Fisher, Queensland
Mr Andrew Wilkie: Member for Denison, Tasmania
```

c. Write an egrep command that will print all the lines in the file where the MP's surname (last name) ends in the letter 'y'. Hint the output shoul

```
Hon Linda Burney: Member for Barton, New South Wales
Mr Pat Conroy: Member for Charlton, New South Wales
Hon Michael Danby: Member for Melbourne Ports, Victoria
Hon David Feeney: Member for Batman, Victoria
Ms Justine Keay: Member for Braddon, Tasmania
Mr Craig Kelly: Member for Hughes, New South Wales
The Hon Dr Mike Kelly AM: Member for Eden-Monaro, New South Wales
Ms Michelle Landry: Member for Capricornia, Queensland
Hon Craig Laundy: Member for Reid, New South Wales
Hon Sussan Ley: Member for Farrer, New South Wales
Mr Rowan Ramsey: Member for Grey, South Australia
Ms Anne Stanley: Member for Werriwa, New South Wales
```

```
$ egrep 'y( [A-Z]*)?:' parliament.txt
Hon Linda Burney: Member for Barton, New South Wales
Mr Pat Conroy: Member for Charlton, New South Wales
Hon Michael Danby: Member for Melbourne Ports, Victoria
Hon David Feeney: Member for Batman, Victoria
Ms Justine Keay: Member for Braddon, Tasmania
Mr Craig Kelly: Member for Hughes, New South Wales
The Hon Dr Mike Kelly AM: Member for Eden-Monaro, New South Wales
Ms Michelle Landry: Member for Capricornia, Queensland
Hon Craig Laundy: Member for Reid, New South Wales
Hon Sussan Ley: Member for Farrer, New South Wales
Mr Rowan Ramsey: Member for Grey, South Australia
Ms Anne Stanley: Member for Werriwa, New South Wales
```

Note this more obvious answer does not handle the MP having an Order of Australia:

```
$ egrep 'y:' parliament.txt
Hon Linda Burney: Member for Barton, New South Wales
Mr Pat Conroy: Member for Charlton, New South Wales
Hon Michael Danby: Member for Melbourne Ports, Victoria
Hon David Feeney: Member for Batman, Victoria
Ms Justine Keay: Member for Braddon, Tasmania
Mr Craig Kelly: Member for Hughes, New South Wales
Ms Michelle Landry: Member for Capricornia, Queensland
Hon Craig Laundy: Member for Reid, New South Wales
Hon Sussan Ley: Member for Farrer, New South Wales
Mr Rowan Ramsey: Member for Grey, South Australia
Ms Anne Stanley: Member for Werriwa, New South Wales
```

d. Write an egrep command that will print all the lines in the file where the MP's name **and** the electorate ends in the letter 'y'. Hint the output sh be:

```
Mr Rowan Ramsey: Member for Grey, South Australia
```

```
$ egrep 'y( [A-Z]*)?:.*y,' parliament.txt
Mr Rowan Ramsey: Member for Grey, South Australia
```

Note this more obvious answer does not handle the MP having an Order of Australia:

```
$ egrep 'y:.*y,' parliament.txt
Mr Rowan Ramsey: Member for Grey, South Australia
```

e. Write an egrep command that will print all the lines in the file where the MP's name **or** the electorate ends in the letter 'y'. Hint the output sho

```
Hon Linda Burney: Member for Barton, New South Wales
Mr Pat Conroy: Member for Charlton, New South Wales
Mr Chris Crewther: Member for Dunkley, Victoria
Hon Michael Danby: Member for Melbourne Ports, Victoria
Mr Milton Dick: Member for Oxley, Queensland
Hon David Feeney: Member for Batman, Victoria
Hon Ed Husic: Member for Chifley, New South Wales
Mr Stephen Jones: Member for Throsby, New South Wales
Hon Bob Katter: Member for Kennedy, Queensland
Ms Justine Keay: Member for Braddon, Tasmania
Mr Craig Kelly: Member for Hughes, New South Wales
The Hon Dr Mike Kelly AM: Member for Eden-Monaro, New South Wales
Ms Michelle Landry: Member for Capricornia, Queensland
Hon Craig Laundy: Member for Reid, New South Wales
Hon Sussan Ley: Member for Farrer, New South Wales
Mr Ben Morton: Member for Tangney, Western Australia
Mr Llew O'Brien: Member for Wide Bay, Queensland
Hon Tanya Plibersek: Member for Sydney, New South Wales
Mr Rowan Ramsey: Member for Grey, South Australia
Ms Michelle Rowland: Member for Greenway, New South Wales
The Hon Tony Smith: Member for Casey, Victoria
Ms Anne Stanley: Member for Werriwa, New South Wales
Hon Wayne Swan: Member for Lilley, Queensland
Mr Trent Zimmerman: Member for North Sydney, New South Wales
```

```
$ egrep 'y( [A-Z]*)?:|y,' parliament.txt
Hon Linda Burney: Member for Barton, New South Wales
Mr Pat Conroy: Member for Charlton, New South Wales
Mr Chris Crewther: Member for Dunkley, Victoria
Hon Michael Danby: Member for Melbourne Ports, Victoria
Mr Milton Dick: Member for Oxley, Queensland
Hon David Feeney: Member for Batman, Victoria
Hon Ed Husic: Member for Chifley, New South Wales
Mr Stephen Jones: Member for Throsby, New South Wales
Hon Bob Katter: Member for Kennedy, Queensland
Ms Justine Keay: Member for Braddon, Tasmania
Mr Craig Kelly: Member for Hughes, New South Wales
The Hon Dr Mike Kelly AM: Member for Eden-Monaro, New South Wales
Ms Michelle Landry: Member for Capricornia, Queensland
Hon Craig Laundy: Member for Reid, New South Wales
Hon Sussan Ley: Member for Farrer, New South Wales
Mr Ben Morton: Member for Tangney, Western Australia
Mr Llew O'Brien: Member for Wide Bay, Queensland
Hon Tanya Plibersek: Member for Sydney, New South Wales
Mr Rowan Ramsey: Member for Grey, South Australia
Ms Michelle Rowland: Member for Greenway, New South Wales
The Hon Tony Smith: Member for Casey, Victoria
Ms Anne Stanley: Member for Werriwa, New South Wales
Hon Wayne Swan: Member for Lilley, Queensland
Mr Trent Zimmerman: Member for North Sydney, New South Wales
```

Note this more obvious answer does not handle the MP having an Order of Australia:

```
$ egrep 'y[:,]' parliament.txt
Hon Linda Burney: Member for Barton, New South Wales
Mr Pat Conroy: Member for Charlton, New South Wales
Mr Chris Crewther: Member for Dunkley, Victoria
Hon Michael Danby: Member for Melbourne Ports, Victoria
Mr Milton Dick: Member for Oxley, Queensland
Hon David Feeney: Member for Batman, Victoria
Hon Ed Husic: Member for Chifley, New South Wales
Mr Stephen Jones: Member for Throsby, New South Wales
Hon Bob Katter: Member for Kennedy, Queensland
Ms Justine Keay: Member for Braddon, Tasmania
Mr Craig Kelly: Member for Hughes, New South Wales
Ms Michelle Landry: Member for Capricornia, Queensland
Hon Craig Laundy: Member for Reid, New South Wales
Hon Sussan Ley: Member for Farrer, New South Wales
Mr Ben Morton: Member for Tangney, Western Australia
Mr Llew O'Brien: Member for Wide Bay, Queensland
Hon Tanya Plibersek: Member for Sydney, New South Wales
Mr Rowan Ramsey: Member for Grey, South Australia
Ms Michelle Rowland: Member for Greenway, New South Wales
The Hon Tony Smith: Member for Casey, Victoria
Ms Anne Stanley: Member for Werriwa, New South Wales
Hon Wayne Swan: Member for Lilley, Queensland
Mr Trent Zimmerman: Member for North Sydney, New South Wales
```

f. Write an egrep command to print all the lines in the file where there is any part of the MP's name or the electorate name that ends in ng. Hint output should be:

```
Mr John Alexander OAM: Member for Bennelong, New South Wales
Hon Josh Frydenberg: Member for Kooyong, Victoria
Mr Luke Gosling: Member for Solomon, Northern Territory
Mr Andrew Hastie: Member for Canning, Western Australia
Hon Michael Keenan: Member for Stirling, Western Australia
Hon Catherine King: Member for Ballarat, Victoria
Ms Madeleine King: Member for Brand, Western Australia
Mr Andrew Laming: Member for Bowman, Queensland
Hon Bill Shorten: Member for Maribyrnong, Victoria
```

```
$ egrep 'ng[^a-z]' parliament.txt
Mr John Alexander OAM: Member for Bennelong, New South Wales
Hon Josh Frydenberg: Member for Kooyong, Victoria
Mr Luke Gosling: Member for Solomon, Northern Territory
Mr Andrew Hastie: Member for Canning, Western Australia
Hon Michael Keenan: Member for Stirling, Western Australia
Hon Catherine King: Member for Ballarat, Victoria
Ms Madeleine King: Member for Brand, Western Australia
Mr Andrew Laming: Member for Bowman, Queensland
Hon Bill Shorten: Member for Maribyrnong, Victoria
```

g. Write an egrep command that will print all the lines in the file where the MP's surname (last name) both begins and ends with a vowel. Hint th output should be:

```
Hon Anthony Albanese: Member for Grayndler, New South Wales
```

```
$ egrep '[AEIOU][^ ]*[aeiou]:' parliament.txt
Hon Anthony Albanese: Member for Grayndler, New South Wales
```

h. Most electorate have names that are a single word, e.g. Warringah, Lyons & Grayndler. A few electorates have multiple word names, for exam Kingsford Smith. Write an egrep command that will print all the lines in the file where the electorate name contains multiple words (separated spaces or hyphens). Hint the output should be:

```
Hon Mark Butler: Member for Port Adelaide, South Australia
Hon Michael Danby: Member for Melbourne Ports, Victoria
Hon Barnaby Joyce: Member for New England, New South Wales
The Hon Dr Mike Kelly AM: Member for Eden-Monaro, New South Wales
Mr Llew O'Brien: Member for Wide Bay, Queensland
Hon Matt Thistlethwaite: Member for Kingsford Smith, New South Wales
Mr Jason Wood: Member for La Trobe, Victoria
Mr Trent Zimmerman: Member for North Sydney, New South Wales
```

```
$ egrep -i 'Member for [a-z]+[ -][a-z]' parliament.txt
Hon Mark Butler: Member for Port Adelaide, South Australia
Hon Michael Danby: Member for Melbourne Ports, Victoria
Hon Barnaby Joyce: Member for New England, New South Wales
The Hon Dr Mike Kelly AM: Member for Eden-Monaro, New South Wales
Mr Llew O'Brien: Member for Wide Bay, Queensland
Hon Matt Thistlethwaite: Member for Kingsford Smith, New South Wales
Mr Jason Wood: Member for La Trobe, Victoria
Mr Trent Zimmerman: Member for North Sydney, New South Wales
```

Pipelining MPs

a. Write a shell pipeline which prints the 8 Australians states & territory in order of the number of MPs they have. It should print only the names
states/territories. It should print them one per line
Hint: check out the Unix filters cut, sort, uniq in the lecture notes.

Hint the output should be:

```
Australian Capital Territory
Northern Territory
Tasmania
South Australia
Western Australia
Queensland
Victoria
New South Wales
```

```
$ cut -d, -f2 parliament.txt|sort | uniq -c |sort -n|cut -c10-
Australian Capital Territory
Northern Territory
Tasmania
South Australia
Western Australia
Queensland
Victoria
New South Wales
```

b. Challenge: The most common first name for an MP is Andrew. Write a shell pipeline which prints the 2nd most common MP first name. It shou
this first name and only this first name.
Hint: check out the Unix filters cut, sort, sed, head, tail & uniq in the lecture notes.

```
$ cut -d: -f1 parliament.txt|sed 's/ [^ ]*$//;s/.* //'|sort|uniq -c|sort -nr|head -3|sed 's/.* //'|head -2|tail -1
Tony
```

Counting Classes

The file /home/cs2041/public_html/lab/sh/enrollments/classes.txt (lab/sh/enrollments/classes.txt) contains a list of all CSE tute/lab classes downloa
from MyUNSW.

a. How many total classes are there?

Each line in the file represents a class. Therefore, the number of classes is equal to the number of lines in the file.

```
$ wc -l classes.txt
314 classes.txt
```

b. Write a pipeline to print how many different courses have classes? Hint: `cut` with the `-f` option will be useful here. Hint: the output of the pi
should be:

```
36
```

Approach: extract just the course codes, then sort them into groups of identical course codes, then compress each group to size one, giving
one line for each group (i.e. each course). Then count the number of lines.

```
$ cut -f1 classes.txt | sort | uniq | wc -l
36
```

c. Write a pipeline which will print the course with the most classes (and no other courses) and how many classes are in this course? Hint: the ou
the pipeline should be:

```
29 ENGG1811
```

```
$ cut -f1 classes.txt | sort | uniq -c|sort -n|tail -1
     29 ENGG1811
```

d. Write a pipeline that prints the most frequently-used tut room and how often it is used? Note you have to include tut-labs. In otherwords TLB a
as TUT. Hint: the output of the pipeline should be:

```
13 Quad G041
```

Approach: extract the tutes and tut-labs, cut out the room names, then sort them into groups, then count the number of entries in each gr
then sort numerically and grab the last line.

```
$ egrep 'TUT|TLB' classes.txt|cut -f5|sort|uniq -c|sort |tail -1
     13 Quad G041
```

The last two steps in the above pipelines could be changed to `sort –nr | head –1`

e. Write a pipeline that prints the most popular time-of-day for tuts (make sure you include tut-labs) and how many tut-labs are at that time? Hi
has a `–c` option. Hint: the output of the pipeline should be:

```
25 12-13
```

```
$  egrep 'TUT|TLB' classes.txt|cut -f4|cut -c 5-9|sort|uniq -c|sort -n|tail -1
     25 12-13
```

The last two steps in the above pipelines could be changed to `sort –nr | head –1`

f. Write a pipeline to discover which COMP courses run the most simultaneous classes of the same type? (e.g. three tutes at the same time on th
same day). This one might take a little thought, but remember that this time we're looking for whether something is duplicated or not, rather
counting unique occurrences of something.

This question needs a little interpretation. The idea is that we're interested in any course where two classes are running on the same day a
same time. The info we want is contained in fields 1, 3 and 4 of each line. If we cut these lines out, then sort the file, then count the size o
each group, we could extract the final result by picking out groups whose size was different to '1'. The following pipeline uses this approach

```
$ egrep '^COMP' classes.txt | cut -f1,3,4|sort|uniq -c|sort|tail
      2 COMP9311        LAB     Wed 19-21
      2 COMP9331        LAB     Mon 12-14
      2 COMP9331        LAB     Mon 18-20
      2 COMP9331        LAB     Thu 12-14
      2 COMP9331        LAB     Tue 12-14
      2 COMP9331        LAB     Tue 14-16
      2 COMP9331        LAB     Wed 15-17
      3 COMP1927        TLB     Tue 12-13
      3 COMP2041        TLB     Thu 18-19
      3 COMP9041        TLB     Thu 18-19
```

g. Challenge: write a pipeline that prints list of the course names (only) of COMP courses that run 2 or more simultaneous classes of the same ty
(e.g. three tutes at the same time on the same day).
Hint this should be the output of your pipeline:

```
COMP1917
COMP1921
COMP1927
COMP2041
COMP3331
COMP6733
COMP6771
COMP9041
COMP9242
COMP9311
COMP9331
```

This question needs a little interpretation. The idea is that we're interested in any course where two classes are running on the same day at same time. The info we want is contained in fields 1, 3 and 4 of each line. If we cut these lines out, then sort the file, then count the size of each group, we could extract the final result by picking out groups whose size was different to '1'. The following pipeline uses this approach

```
$ egrep '^COMP' classes.txt | cut -f1,3,4|sort|uniq -c|grep -v '^ *1 '|cut -c9-17|sort|uniq
COMP1917
COMP1921
COMP1927
COMP2041
COMP3331
COMP6733
COMP6771
COMP9041
COMP9242
COMP9311
COMP9331
```

or

```
$ egrep '^COMP' classes.txt | cut -f1,3,4|sort|uniq -d|cut -f1|sort|uniq
COMP1917
COMP1921
COMP1927
COMP2041
COMP3331
COMP6733
COMP6771
COMP9041
COMP9242
COMP9311
COMP9331
```

Challenge: Interesting Regexps

Use `egrep` to test your answers to these questions.
Try to solve these questions using the standard regular expression language described in lectures.

a. Write a regular expression for egrep that matches any line containing at least one `A` and at least one `B`. For example:

Matching	**Not Matching**
Andrew's favourite Band is not	George is Brillant
ABBA	Andrew
BA	B
AB	A

So to test with egrep you might do this:

```
% cat >file1 <<eof
Andrew's favourite Band is not
George is Brillant
ABBA
Andrew
AB
BA
A
B
eof
% egrep 'REGEXP' <file1
Andrew's favourite Band is not
ABBA
AB
BA
```

```
A.*B|B.*A
```

b. Write a regular expression for egrep that matches any line containing only the characters `A` and `B` such that all pairs of adjacent A's occur be any pairs of adjacent B's. In other words if there is pair of B's on the line , there can not be a pair of A's afterwards.

Matching	**Not Matching**
ABAABAABAABBBBABB	BBAA
ABBA	ABBAA
ABAAAAAAAAAABBA	ABBABABABABAA
ABABABABA	ABBBAAA
A	BBABABABABABABAA

```
^(BA|A)*(|B)(AB|B)*(|A)$
```

or courtesy Squish:

```
^(BA|A)*(BA|B)*$
```

c. Write a regular expression for egrep that matches any line containing only the characters `A` and `B` such that the number of A's is divisible by

Matching	**Not Matching**
AAAA	AAAAA
BABABABAB	ABABBBBBBBBBBBBBBBAAA
AAAABBBBAAAA	AAAABBABBAAAA
BBBAABBBBBAABBBAAAA	BBBAABBABBBAABBBAAAA

```
^B*(AB*AB*AB*AB*)*$
```

d. Write a regular expression for egrep that matches any line containing only the characters `A` and `B` such that there are exactly *n* A's followed b exactly *n* B's and no other characters.

Matching	**Not Matching**
AAABBB	AAABB
AB	BA
AABB	AABBB
AAAABBBB	AAAABBBBA

If you can't invent a regular expression, can you write a shell script using egrep, sed and test, if & while which performs the same task.

This can't be done with a (true) regular expression. You prove this via the the wonderfully named pumping lemma (http://en.wikipedia.org/wiki/Pumping_lemma_for_regular_languages).
It is possible with extensions to the regular expression language provided in languages such as Perl (http://faq.perl.org/perlfaq6.html#Can_I_use_Perl_regulextension).

Shell script which prints line containing n As followed by n Bs<

```
#!/bin/sh
while read line
do
    string="$line"
    while test -n "`echo "$string"|egrep '^A.*B$'`"
    do
        string=`echo "$string"|sed 's/^A//'`
        string=`echo "$string"|sed 's/B$//'`
    done
    test -z "$string" && echo "$line"
done
```

Here is a clever solution, courtesy Alexis Shaw, using grep, and some features of sed we don't cover:

Grep & sed script which prints line containing n As followed by n Bs

```
#!/bin/sh
# written by Alexis Shaw
egrep "^A*B*$"|
sed "s/\(A*\)\(B*\)/\1|\2/;s/B/A/g"|
sed -n "s/^\(A*\)|\1$/\1|\1/p"|
sed -n "s/^\(A*\)|\(A*\)$/\1\n|\1/gp"|
sed "/|/ s/A/B/g"|
sed "N; s/\n|//"
```

Here is more sed-ish solution, these sed feature aren't covered in the course and languages such as Perl & Python are generally used instea these days:

Sed script which prints line containing n As followed by n Bs

```
#!/bin/sh
sed '
    /^A*B*$/!d
    h
    s/\(B*\)$/C\1/
    s/B/A/g
    /^\(A*\)C\1$/!d
    g
'
```

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 lab02 lab02.txt`. Whether you discuss your solutions with your tutor this week or next week, you must submit them before the deadline.