

Objectives

- Understanding regular expressions
- Understanding use of UNIX filters

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 **lab01** by typing:

```
$ mkdir lab01
```

Change to this directory by typing:

```
$ cd lab01
```

Exercise: egrep-ing a Dictionary with egrep (individual)

This is an individual exercise to complete by yourself.

There is a template file named **dictionary_answers.txt** which you must use to enter the answers for this exercise. Download **dictionary_answers.txt** [here](#), or copy it to your CSE account using the following command:

```
$ cp -n /web/cs2041/19T2/activities/dictionary/dictionary_answers.txt .
```

The autotest scripts depend on the format of **dictionary_answers.txt** so just add your answers don't otherwise change the file. In other words edit **dictionary_answers.txt**:

```
$ gedit dictionary_answers.txt &
```

On most Unix systems you will find one or more dictionaries containing many thousands of words typically in the directories **/usr/share/dict/**.

We've created a dictionary named **dictionary.txt** for this lab exercise.

Download **dictionary.txt** [here](#), or copy it to your CSE account using the following command:

```
$ cp -n /web/cs2041/19T2/activities/dictionary/dictionary.txt .
```

```
$ ls -l
total 4
lrwxrwxrwx 1 cs2041 cs2041 17 May 30 17:42 dictionary.txt -> ../dic
tionary.txt
-rw-r--r-- 1 cs2041 cs2041 1072 May 30 17:42 dictionary_answers.txt
```

1. Write an egrep command that prints the words in **dictionary.txt** which contain in characters "lmn" consecutively.
Hint: it should print:

```
Selmner
Selmner's
almner
almners
calmness
calmness's
calmnesses
```

Sample answer:

```
egrep 'lmn' dictionary.txt
```

The COMP2041 class account contains a script named **autotest** that automatically runs 1 or more tests on your lab exercises. Once you have entered your answer for q1 you can check it like this:

```
$ 2041 autotest dictionary q1
Test q1 (egrep '^Q1 answer' dictionary_answers.txt|tail -1|sed
's/.*answer[: ]*//'|sh) - passed
1 tests passed 0 tests failed
```

Passing the autotest doesn't guarantee your answer is correct, of course, but it may find a mistake you've missed so run autotest for each of the following questions when you've entered the answer in **dictionary_answers.txt**.

- Write a shell pipeline that prints the words that contain "zz", but do not end in apostrophe-s ('s)?

Hint: it should print:

```
Abruzzi
Arezzini
Arezzo
Barozzi
Belshazzar
Brazzaville
Buzz
Buzzell
Cazzie
Chuzzlovit
```

Sample answer:

```
egrep 'zz' dictionary.txt | egrep -v \"'s$\"
```

- Write an egrep command that prints the words that contain four consecutive vowels?

Hint: it should print these words:

```
Aiea
Aiea's
Araguaia
Araguaia's
Douai
Douai's
Graeae
Graiae
Hawaiian
Hawaiians
```

Sample answer:

```
egrep '[aeiouAEIOU][aeiouAEIOU][aeiouAEIOU][aeiouAEIOU]' dictionary.txt
```

or using egrep's -i option to ignore case and {} for repetition:

```
egrep -i '[aeiou]{4}' dictionary.txt
```

4. Write an egrep command that prints words which contain all 5 vowels "aeiou" in that order?
The words may contain more than 5 vowels but they must contain "aeiou" in that order.

Hint: it should print these words:

```
abstemious
abstemiously
abstemiousness
abstemiousness's
abstemiousnesses
abstentious
adenocarcinomatous
adventitious
adventitiously
adventitiousness
```

Sample answer:

```
egrep -i 'a.*e.*i.*o.*u' dictionary.txt
```

5. Write an egrep command that prints which contain the vowels "aeiou" in that order and no other vowels.

Hint: it should print these words:

```
abstemious
abstemiously
abstentious
arsenious
caesious
facetious
facetiously
```

Sample answer:

```
egrep -i '^([aeiou]*a[aeiou]*e[aeiou]*i[aeiou]*o[aeiou]*u[aeiou]*$' dictionary.txt
```

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

```
$ 2041 autotest dictionary
```

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

```
$ give cs2041 lab01_dictionary dictionary_answers.txt
```

You must run give before **Tuesday 11 June 17:59:59** to obtain the marks for this lab exercise. Note, this is an individual exercise, the work you submit with **give** must be entirely your own.

Sample solution for **dictionary_answers.txt**

This file is automarked.

Do not add extra lines to this file, just add your answers.

For example **if** your answer to Q1 is: `egrep Andrew words.txt`
Change the line that says Q1 answer to:

Q1 answer: `egrep Andrew words.txt`

1) Write an egrep command that prints the words **in** dictionary.txt which contain **in** characters `"lmn"` consecutively.

Q1 answer: `egrep 'lmn' dictionary.txt`

2) Write a shell pipeline that prints the words that contain `"zz"`, but **do** not end **in** apostrophe-s (')

Q2 answer: `egrep 'zz' dictionary.txt | egrep -v "'s$"`

3) Write an egrep command that prints the words that contain four consecutive vowels?

Q3 answer: `egrep '[aeiouAEIOU][aeiouAEIOU][aeiouAEIOU][aeiouAEIOU]' dictionary.txt`

4) Write an egrep command that prints words which contain all 5 vowels `"aeiou"` **in** that order?

Q4 answer: `egrep -i 'a.*e.*i.*o.*u' dictionary.txt`

5) Write an egrep command that prints which contain the vowels `"aeiou"` **in** that order and no other vowels.

Q5 answer: `egrep -i '^^[^aeiou]*a[^aeiou]*e[^aeiou]*i[^aeiou]*o[^aeiou]*u[^aeiou]*$' dictionary.txt`

Exercise: egrep-ing Federal Parliament (individual)

This is an individual exercise to complete by yourself.

There is a template file named `parliament_answers.txt` which you must use to enter the answers for this exercise.
Download `parliament_answers.txt` [here](#), or copy it to your CSE account using the following command:

```
$ cp -n /web/cs2041/19T2/activities/parliament/parliament_answers.txt .
```

The autotest scripts depend on the format of `parliament_answers.txt` so just add your answers don't otherwise change the file.

In this exercise you will analyze a file named `parliament.txt` containing a list of the members of the Australian House of Representatives (MPs).

Download `parliament.txt` [here](#), or copy it to your CSE account using the following command:

```
$ cp -n /web/cs2041/19T2/activities/parliament/parliament.txt .
```

1. Write an egrep command that will print all the lines in the file where the electorate begins with W.
Hint: it should print these lines:

Hon Scott Buchholz: Member for Wright, Queensland
 Hon Tony Burke: Member for Watson, New South Wales
 Mr Nick Champion: Member for Wakefield, South Australia
 Mr Stephen Jones: Member for Whitlam, New South Wales
 Mr Peter Khalil: Member for Wills, Victoria
 Mr Llew O'Brien: Member for Wide Bay, Queensland
 Dr Kerry Phelp AM: Member for Wentworth, New South Wales
 Ms Anne Stanley: Member for Werriwa, New South Wales
 Ms Zali Steggall OAM: Member for Warringah, New South Wales
 Hon Dan Tehan: Member for Wannon, Victoria

Sample answer:

```
egrep 'Member for W' parliament.txt
```

2. Write an egrep command that will list all the lines in the file where the MP's first name is Andrew.

Hint: it should print these words:

Hon 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

Sample answer:

```
egrep ' Andrew .*:' parliament.txt
```

This depends on Andrew not being the surname of someone with an honorific such as OAM after their name.

3. Write an egrep command that will print all the lines in the file where the MP's surname (last name) ends in the letters 'll'.

Hint: it should print these words:

Mr Julian Hill: Member for Bruce, Victoria
 Mr Brian Mitchell: Member for Lyons, Tasmania
 Mr Rob Mitchell: Member for McEwen, Victoria
 Ms Zali Steggall OAM: Member for Warringah, New South Wales

Sample answer:

```
egrep 'll( [A-Z]*)?:' parliament.txt
```

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

```
egrep 'll:' parliament.txt
```

4. 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: it should print these lines:

Mr Rowan Ramsey: Member for Grey, South Australia

Sample answer:

```
egrep 'y( [A-Z]*)?:.*y,' parliament.txt
```

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

```
egrep 'y:.*y,' parliament.txt
```

5. 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: it should print these lines:

```
Dr Anne Aly: Member for Cowan, Western Australia
Hon Linda Burney: Member for Barton, New South Wales
Mr Pat Conroy: Member for Shortland, New South Wales
Mr Chris Crewther: Member for Dunkley, Victoria
Mr Milton Dick: Member for Oxley, Queensland
Hon Damian Drum: Member for Murray, Victoria
Ms Nicolle Flint: Member for Boothby, South Australia
Hon Ed Husic: Member for Chifley, New South Wales
Hon Bob Katter: Member for Kennedy, Queensland
Ms Ged Kearney: Member for Batman, Victoria
Mr Craig Kelly: Member for Hughes, New South Wales
Hon Dr Mike Kelly AM: Member for Eden-Monaro, New South Wales
Hon Michelle Landry: Member for Capricornia, Queensland
Hon Sussan Ley: Member for Farrer, New South Wales
Mrs Melissa McIntosh: Member for Lindsay, New South Wales
Hon 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
Hon Tony Smith: Member for Casey, Victoria
Ms Anne Stanley: Member for Werriwa, New South Wales
Mr Trent Zimmerman: Member for North Sydney, New South Wales
```

Sample answer:

```
egrep 'y( [A-Z]*)?:|y,' parliament.txt
```

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

```
egrep 'y[:|,]' parliament.txt
```

6. Write an egrep command to print all the lines in the file where there is any word in the MP's name or the electorate name that ends in ng.

Hint: it should print these lines:

```
Mr John Alexander OAM: Member for Bennelong, New South Wales
Hon Josh Frydenberg: Member for Kooyong, Victoria
Mr Luke Gosling OAM: Member for Solomon, Northern Territory
Mr Andrew Hastie: Member for Canning, 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
Mr Terry Young: Member for Longman, Queensland
```

Sample answer:

```
egrep 'ng[^a-z]' parliament.txt
```

7. 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: it should print these lines:

```
Hon Anthony Albanese: Member for Grayndler, New South Wales
Ms Cathy O'Toole: Member for Herbert, Queensland
```

Sample answer:

```
egrep '[AEIOU][^ ]*[aeiou]([A-Z]*)?:' parliament.txt
```

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

Hint: it should print these lines:

```
Hon Mark Butler: Member for Port Adelaide, South Australia
Hon Barnaby Joyce: Member for New England, New South Wales
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 Wal
es
Hon Jason Wood: Member for La Trobe, Victoria
Mr Trent Zimmerman: Member for North Sydney, New South Wales
```

Sample answer:

```
egrep -i 'Member for [a-z]+[ -][a-z]' parliament.txt
```

9. Write a shell pipeline which prints the 8 Australian states & territory in order of the number of MPs they have. It should print only the number of MPs, followed by the name of the states/territories. It should print them from fewest to most MPs. Hint: check out the Unix filters cut, sort, uniq in the lecture notes.

Hint: it should print these lines:

```
1 Australian Capital Territory
2 Northern Territory
5 Tasmania
9 South Australia
15 Western Australia
27 Queensland
33 Victoria
45 New South Wales
```

Sample answer:

```
cut -d: -f2 parliament.txt|cut -d, -f2|sort|uniq -c|sort -n
```

10. 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 should print this first name and only this first name.

Hint: check out the Unix filters cut, sort, sed, head, tail & uniq in the lecture notes.

Hint: it should print this line:

```
Tony
```

Sample answer:

```
cut -d: -f1 parliament.txt|sed 's/ Dr / /'|cut -d' ' -f2|sort|uniq -c|sort|sed 's/.* //'|tail -2|head -1
```

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

```
$ 2041 autotest parliament
```

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

```
$ give cs2041 lab01_parliament parliament_answers.txt
```

You must run give before **Tuesday 11 June 17:59:59** to obtain the marks for this lab exercise. Note, this is an individual exercise, the work you submit with **give** must be entirely your own.

Sample solution for `parliament_answers.txt`

This file is automarked.

Do not add extra lines to this file, just add your answers.

For example `if` your answer to Q1 is: `egrep Andrew words.txt`
Change the line that says Q1 answer to:

Q1 answer: `egrep Andrew words.txt`

1) Write an `egrep` command that will print all the lines `in` the file where the electorate begins with

Q1 answer: `egrep 'Member for W' parliament.txt`

2) Write an `egrep` command that will list all the lines `in` the file where the MP's first name is Andrew.

Q2 answer: `egrep ' Andrew .*:' parliament.txt`

3) Write an `egrep` command that will print all the lines `in` the file where the MP's surname (last name) ends `in` the letters `'ll'`.

Q3 answer: `egrep 'll([A-Z]*)?:' parliament.txt`

4) 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'`.

Q4 answer: `egrep 'y([A-Z]*)?:.*y,' parliament.txt`

5) 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'`.

Q5 answer: `egrep 'y([A-Z]*)?:ly,' parliament.txt`

6) Write an `egrep` command to print all the lines `in` the file where there is any word `in` the MP's name or the electorate name that ends `in` `ng`.

Q6 answer: `egrep 'ng^[^a-z]'` parliament.txt

7) 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.

Q7 answer: `egrep '[AEIOU][^]*[aeiou]([A-Z]*)?:' parliament.txt`

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

Q8 answer: `egrep -i 'Member for [a-z]+[-][a-z]'` parliament.txt

9) Write a shell pipeline which prints the 8 Australian states & territory `in` order of the number of MPs. It should print only the number of MPs, followed by the name of the states/territories. It should print them from fewest to most MPs.

Q9 answer: `cut -d: -f2 parliament.txt|cut -d, -f2|sort|uniq -c|sort -n`

10) 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 should print this first name and only this first name.

Q10 answer: `cut -d: -f1 parliament.txt|sed 's/ Dr / /'|cut -d' ' -f2|sort|uniq -c|sort|sed 's/.* //'|`

Challenge Exercise: Exploring Regular Expressions (individual)

This is an individual exercise to complete by yourself.

There is a template file named `ab_answers.txt` which you must use to enter the answers for this exercise. Download `ab_answers.txt` [here](#), or copy it to your CSE account using the following command:

```
$ cp -n /web/cs2041/19T2/activities/ab/ab_answers.txt .
```

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

1. Write a `egrep` command that prints the lines in a file named `input.txt` containing containing at least one **A** and at least one **B**. For example:

| Matching | Not Matching |
|--------------------------------|---------------------|
| Andrew's favourite Band is not | George is Brilliant |
| ABBA | Andrew |
| BA | B |
| AB | A |

So to test with `egrep` you might do this:

```
$ cat >input.txt <<eof
Andrew's favourite Band is not
George is Brilliant
ABBA
Andrew
AB
BA
A
B
eof
$ egrep 'REGEXP' input.txt
Andrew's favourite Band is not
ABBA
AB
BA
```

```
egrep 'A.*B|B.*A' input.txt
```

2. Write a `egrep` command that prints the lines in a file named `input.txt` containing only the characters **A** and **B** such that all pairs of adjacent A's occur before 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 |

```
egrep '^ (BA|A)* (|B) (AB|B)* (|A)$' input.txt
```

or courtesy Squish:

```
egrep '^ (BA|A)* (BA|B)*$' input.txt
```

3. Write a egrep command that prints the lines in a file named `input.txt` containing only the characters **A** and **B** such that the number of A's is divisible by 4.

| Matching | Not Matching |
|---------------------|-----------------------|
| AAAA | AAAAA |
| BABABABAB | ABABBBBBBBBBBBBBBAAA |
| AAAABBBBAAAA | AAAABBABBAAAA |
| BBBAABBBBBAABBBAAAA | BBBAABBABBBBAABBBAAAA |

```
egrep '^B*(AB*AB*AB*AB*)*$' input.txt
```

4. Write a egrep command that prints the lines in a file named `input.txt` containing only the characters **A** and **B** such that there are exactly n A's followed by exactly n B's and no other characters.

| Matching | Not Matching |
|----------|--------------|
| AAABBB | AAABB |
| AB | BA |
| AABB | AABBB |
| AAAABBBB | AAAABBBBA |

This can't be done with a (true) regular expression. You prove this via the the wonderfully named [pumping lemma](#). It is possible with extensions to regular expressions, e.g. as provided in Perl.

```
grep -P '^ (A(?1)?B)$' input.txt
```

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

```
$ 2041 autotest ab
```

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

```
$ give cs2041 lab01_ab ab_answers.txt
```

You must run give before **Tuesday 11 June 17:59:59** to obtain the marks for this lab exercise. Note, this is an individual exercise, the work you submit with **give** must be entirely your own.

Sample solution for `ab_answers.txt`

This file is automarked.

Do not add extra lines to this file, just add your answers.

For example `if` your answer to Q1 is: `egrep Andrew words.txt`
Change the line that says Q1 answer to:

Q1 answer: `egrep Andrew words.txt`

1) Write a `egrep` command that prints the lines `in` a file named `input.txt` containing containing at least one A and at least one B.

Q1 answer: `egrep 'A.*BIB.*A' input.txt`

2) Write a `egrep` command that prints the lines `in` a file named `input.txt` containing only the characters A and B such that all pairs of adjacent A's occur before 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.

Q2 answer: `egrep '^ (BAIA)* (IB) (ABIB)* (IA)$' input.txt`

3) Write a `egrep` command that prints the lines `in` a file named `input.txt` containing only the characters A and B such that the number of A's is divisible by 4.

Q3 answer: `egrep '^B* (AB*AB*AB*AB*)*$' input.txt`

4) Write a `egrep` command that prints the lines `in` a file named `input.txt` containing only the characters A and B such that there are exactly n A's followed by exactly n B's and no other characters.

Q4 answer: `grep -P '^ (A(?1)?B)$' input.txt`

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 **Tuesday 11 June 17:59:59** 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.

COMP(2041|9044) 19T2: Software Construction is brought to you by the [School of Computer Science and Engineering](#) at the [University of New South Wales](#), Sydney.

For all enquiries, please email the class account at cs2041@cse.unsw.edu.au

CRICOS Provider 00098G