Week 01 Laboratory Sample Solutions

Objectives

- Understanding regular expressions
- Understanding use of UNIX filters (grep)

Preparation

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

Getting Started

Set up for the lab by creating a new directory called `lab01` and changing to this directory.

```
$ mkdir lab01
$ cd lab01
```

There are some provided files for this lab which you can fetch with this command:

```
$ 2041 fetch lab01
```

If you're not working at CSE, you can download the provided files as a zip file or a tar file.

EXERCISE:
grep-ing a Dictionary

You have been given a file named `dictionary_answers.txt` .
Which you must use to enter the answers for this exercise.

The autotest scripts depend on the format of `dictionary_answers.txt` .
So just add your answers where indicated but don't otherwise change the file.

```
    # Open a text editor (gedit) in the background (&) and not owned by the current terminal (disown)
    $ gedit dictionary_answers.txt & disown
    # Or use any other text editor of your choosing
```

On most Unix systems you will find one or more dictionaries containing many thousands of words:
Typically in the directory `/usr/share/dict/`

```
    $ ls -1 /usr/share/dict/
    american-english
    american-english-huge
    american-english-insane
    american-english-large
    american-english-small
    british-english
    british-english-huge
    british-english-insane
    british-english-large
    british-english-small
    cracklib-small
    words -> /etc/dictionaries-common/words -> /usr/share/dict/american-english
```

We've created an example dictionary named `dictionary.txt` for this lab exercise.

1. Write a `grep -E` command that prints the words which contain the characters "lmn" consecutively.

HINT:

It should print:

```
almner
almners
calmness
calmnesses
```

ANSWER:

Sample answer:

```
$ grep -E 'lmn' dictionary.txt
```

The COMP2041 class account contains a script named **autotest** that automatically runs tests on your lab exercises.

Once you have entered you answer you can check it like this:

```
$ 2041 autotest dictionary Q1
Test Q1 (dictionary Q1) - passed
1 tests passed 0 tests failed
```

2. Write a `grep -E` command that prints the words which contain any four consecutive vowels.

   HINT:

   It should print:

   ```
   aqueous
   archaeoastronomer
   archaeoastronomers
   archaeoastronomical
   archaeoastronomies
   archaeoastronomy
   banlieue
   beauish
   ```

   ANSWER:

   Sample answer:

   ```
   $ grep -E -i '[aeiou]{4}' dictionary.txt
   ```

   Once you have entered you answer you can check it like this:

   ```
   $ 2041 autotest dictionary Q2
   Test Q2 (dictionary Q2) - passed
   1 tests passed 0 tests failed
   ```

3. Write a `grep -E` command that prints the 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:

   ```
   abstemious
   abstemiously
   abstemiousness
   abstemiousnesses
   abstentious
   adenocarcinomatous
   adventitious
   adventitiously
   ```

ANSWER:

Sample answer:

```
		$ grep -E -i 'a.*e.*i.*o.*u' dictionary.txt
```

Once you have entered you answer you can check it like this:

```
		$ 2041 autotest dictionary Q3
		Test Q3 (dictionary Q3) - passed
		1 tests passed 0 tests failed
```

4. Write a `grep -E` command that prints the words which contain the vowels "aeiou", in that order, and no other vowels.

HINT:

It should print:

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

ANSWER:

Sample answer:

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

Once you have entered you answer you can check it like this:

```
		$ 2041 autotest dictionary Q4
		Test Q4 (dictionary Q4) - passed
		1 tests passed 0 tests failed
```

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
```

before **Tuesday 13 June 12:00 (midday)** (2023-06-13 12:00:00) to obtain the marks for this lab exercise.

SOLUTION:

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 Q0 is: "grep -E Andrew words.txt"
Change the line that starts with
    "Q0 answer:"
to
    "Q0 answer: grep -E Andrew words.txt"
----------------------------------------------------------------------------------------------------
--------------------



1) Write an grep -E command that prints the words which contain the characters "lmn" consecutively.
Q1 answer: grep -E 'lmn' dictionary.txt

# Simply look for the substring 'lmn'
# This is effectively what the default behaviour of grep is so nothing fancy needs to be done here.
# The `-E` flag isn't even needed here, and in fact `-F` would be better as it is faster.


2) Write an grep -E command that prints the words which contain any four consecutive vowels.
Q2 answer: grep -E -i '[aeiou]{4}' dictionary.txt

# The `-i` flag is used to make the search case insensitive.
# Alternatively both the uppercase and lowercase vowels could be specified in the regex.
# '[aeiou]' is a character class that matches any vowel
# '{4}' is a quantifier that matches the previous character class exactly 4 times.


3) Write an grep -E command that prints the words which contain all 5 vowels "aeiou" in that order.
Q3 answer: grep -E -i 'a.*e.*i.*o.*u' dictionary.txt

# The `-i` flag is used to make the search case insensitive.
# Alternatively both the uppercase and lowercase vowels could be specified in the regex.
# '.' is a wildcard that matches any character
# '*' is a quantifier that matches the previous character 0 or more times.
# '.*' is a very *very* common regex pattern that matches any string (any number of any characters)
# 'a.*e' would match any string that first contains an 'a' then any number of any characters then an
'e'
# Repeating this pattern for each vowel ensures that the vowels are in the correct order.


4) Write an grep -E command that prints the words which contain the vowels "aeiou", in that order,
and no other vowels.
Q4 answer: grep -E -i '^[^aeiou]*a[^aeiou]*e[^aeiou]*i[^aeiou]*o[^aeiou]*u[^aeiou]*$' dictionary.txt

# The `-i` flag is used to make the search case insensitive.
# '^' is an anchor that matches the start of a line
# '$' is an anchor that matches the end of a line
# '[^aeiou]' is a character class that matches any character that is *not* a vowel
# If `^` if the first thing in a character class it negates the class, so '[^aeiou]' matches the
opposite of '[aeiou]'
# This pattern works by replacing the `.*` in the previous question with `[^aeiou]*`
# This ensures that the only non-vowel characters can be between the vowels.
# adding additional `[^aeiou]*` to the start and end ensures that there are no vowels anywhere in
the word except where we want them.
```

EXERCISE:

grep-ing Federal Parliament

You have been given a file named `parliament_answers.txt` .
Which you must use to enter the answers for this exercise.

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

```
# Open a text editor (gedit) in the background (&) and not owned by the current terminal (disown)
$ gedit parliament_answers.txt & disown
# Or use any other text editor of your choosing
```

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

NOTE:

As we have just had an election the information in the file `parliament.txt` might not be up to date.

1. Write a `grep -E` command that will print all the lines in the file where the electorate begins with 'W'.

   HINT:

   It should print:

   ```
   Hon Scott Buchholz: Member for Wright, Queensland
   Hon Tony Burke: Member for Watson, New South Wales
   Hon Stephen Jones: Member for Whitlam, New South Wales
   Mr Peter Khalil: Member for Wills, Victoria
   Mr Llew O'Brien: Member for Wide Bay, Queensland
   Ms Allegra Spender: 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
   ```

   ANSWER:

   Sample answer:

   ```
   $ grep -E 'Member for W' parliament.txt
   ```

   Once you have entered you answer you can check it like this:

   ```
   $ 2041 autotest parliament Q1
   Test Q1 (parliament Q1) - passed
   1 tests passed 0 tests failed
   ```

2. Write a `grep -E` command that will print all the lines in the file where the MP's given name (first name) is "Andrew".

   HINT:

   It should print:

   ```
   Dr Andrew Charlton: Member for Parramatta, New South Wales
   Hon Andrew Gee: Member for Calare, New South Wales
   Hon Andrew Giles: Member for Scullin, Victoria
   Hon Andrew Hastie: Member for Canning, Western Australia
   Hon Dr Andrew Leigh: Member for Fenner, Australian Capital Territory
   Mr Andrew Wallace: Member for Fisher, Queensland
   Mr Andrew Wilkie: Member for Clark, Tasmania
   Mr Andrew Willcox: Member for Dawson, Queensland
   ```

   ANSWER:

   Sample answer:

   ```
   $ grep -E '^((Mr|Mrs|Ms|Dr|Hon) )*Andrew .*:' parliament.txt
   ```

   Note this more obvious answer will also match middle names

   ```
   $ grep -E ' Andrew .*:' parliament.txt
   ```

Once you have entered you answer you can check it like this:

```
$ 2041 autotest parliament Q2
Test Q2 (parliament Q2) - passed
1 tests passed 0 tests failed
```

3. Write a `grep -E` 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:

```
Ms Angie Bell: Member for Moncrieff, Queensland
Mr Sam Birrell: Member for Nicholls, Victoria
Mr Matt Burnell: Member for Spence, South Australia
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
```

ANSWER:

Sample answer:

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

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

```
$ grep -E 'll:' parliament.txt
```

Once you have entered you answer you can check it like this:

```
$ 2041 autotest parliament Q3
Test Q3 (parliament Q3) - passed
1 tests passed 0 tests failed
```

4. Write a `grep -E` command that will print all the lines in the file where the MP's surname (last name) **and** the electorate name ends in the letter 'y'.

HINT:

It should print:

```
Ms Peta Murphy: Member for Dunkley, Victoria
Mr Rowan Ramsey: Member for Grey, South Australia
```

ANSWER:

Sample answer:

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

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

```
$ grep -E 'y:.*y,' parliament.txt
```

Once you have entered you answer you can check it like this:

```
$ 2041 autotest parliament Q4
Test Q4 (parliament Q4) - passed
1 tests passed 0 tests failed
```

5. Write a `grep -E` command that will print all the lines in the file where the MP's surname (last name) **or** the electorate name ends in the letter 'y'.

HINT:

It should print:

```
Hon Dr Anne Aly: Member for Cowan, Western Australia
Hon Linda Burney: Member for Barton, New South Wales
Ms Kate Chaney: Member for Curtin, Western Australia
Hon Pat Conroy: Member for Shortland, New South Wales
Hon Milton Dick: Member for Oxley, Queensland
Hon Ed Husic: Member for Chifley, New South Wales
Hon Bob Katter: Member for Kennedy, Queensland
Hon Ged Kearney: Member for Cooper, Victoria
Hon Michelle Landry: Member for Capricornia, Queensland
Hon Sussan Ley: Member for Farrer, New South Wales
Mr Sam Lim: Member for Tangney, Western Australia
Mrs Melissa McIntosh: Member for Lindsay, New South Wales
Ms Louise Miller-Frost: Member for Boothby, South Australia
Ms Peta Murphy: Member for Dunkley, Victoria
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
Hon Michelle Rowland: Member for Greenway, New South Wales
Ms Anne Stanley: Member for Werriwa, New South Wales
Ms Kylea Tink: Member for North Sydney, New South Wales
Mr Aaron Violi: Member for Casey, Victoria
```

ANSWER:

Sample answer:

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

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

```
$ grep -E 'y[:,]' parliament.txt
```

Once you have entered you answer you can check it like this:

```
$ 2041 autotest parliament Q5
Test Q5 (parliament Q5) - passed
1 tests passed 0 tests failed
```

6. Write a `grep -E` command that will 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:

```
Mr Luke Gosling OAM: Member for Solomon, Northern Territory
Hon Andrew Hastie: Member for Canning, Western Australia
Hon Catherine King: Member for Ballarat, Victoria
Hon Madeleine King: Member for Brand, Western Australia
Mr Jerome Laxale: Member for Bennelong, New South Wales
Dr Monique Ryan: Member for Kooyong, Victoria
Hon Bill Shorten: Member for Maribyrnong, Victoria
Mr Terry Young: Member for Longman, Queensland
```

ANSWER:

Sample answer:

```
$ grep -E 'ng[^a-z]' parliament.txt
```

Once you have entered you answer you can check it like this:

```
$ 2041 autotest parliament Q6
Test Q6 (parliament Q6) - passed
1 tests passed 0 tests failed
```

7. Write a `grep -E` 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:

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

   ANSWER:

   Sample answer:

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

   Once you have entered you answer you can check it like this:

   ```
   $ 2041 autotest parliament Q7
   Test Q7 (parliament Q7) - passed
   1 tests passed 0 tests failed
   ```

8. Write a `grep -E` 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:

   ```
   Hon Barnaby Joyce: Member for New England, New South Wales
   Hon Kristy McBain: 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
   Ms Kylea Tink: Member for North Sydney, New South Wales
   Hon Jason Wood: Member for La Trobe, Victoria
   ```

   ANSWER:

   Sample answer:

   ```
   $ grep -E 'Member for [a-zA-Z]+[ -][a-zA-Z]' parliament.txt
   ```

   Once you have entered you answer you can check it like this:

   ```
   $ 2041 autotest parliament Q8
   Test Q8 (parliament Q8) - passed
   1 tests passed 0 tests failed
   ```

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
```

before **Tuesday 13 June 12:00 (midday)** (2023-06-13 12:00:00) to obtain the marks for this lab exercise.

SOLUTION:

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 Q0 is: "grep -E Andrew words.txt"
Change the line that starts with
    "Q0 answer:"
to
    "Q0 answer: grep -E Andrew words.txt"
----------------------------------------------------------------------------------------------------
--------------------



1) Write a grep -E command that will print all the lines in the file where the electorate begins
with 'W'.
Q1 answer: grep -E ': Member for W' parliament.txt

# This is just a simple text match
# grep -F, would work just fine here
# We are again using the fact that grep matches a sub-string so we don't need to worry about the
rest of the line
# The `: ` at the start of the pattern is to avoid matching any MPs named "Member for W" (unlikely
but possible)


2) Write a grep -E command that will print all the lines in the file where the MP's first name is
"Andrew".
Q2 answer: grep -E '^((Mr|Miss|Mrs|Ms|Mx|Dr|Hon) )*Andrew' parliament.txt

# `^` matches the start of the line
# We then make the assumption that only the honorifics Mr|Miss|Mrs|Ms|Mx|Dr|Hon will appear.
# The are an awful lot of honorifics, but we can't just use [A-Z][a-z]* because we need to
distinguish between an honorific and a first name
# This list is a fairly safe list to use, but it is not exhaustive
# `|` is the OR operator it allows us to match either the left or right hand side of the expression,
chaining them together allows us to construct a list of possible matches
# `()` are used to group expressions together, in this case we are grouping the honorifics together
so that the `*` applies to all of them
# `*` matches 0 or more of the previous expression, in this case it matches 0 or more honorifics
# `?` could be used to match 0 or 1 of the previous expression, in this case it would match 0 or 1
honorifics
# But some MPs have multiple honorifics, notably "Hon Dr" so we need to match 0 or more

3) Write a grep -E command that will print all the lines in the file where the MP's surname (last
name) ends in the letters 'll'.
Q3 answer: grep -E 'll( [A-Z]+)?:' parliament.txt

# The `:` at the end of the pattern is to avoid matching any electorate with `ll` in the name
# as the `:` is at the end of the pattern the rest of the pattern must match before the `:` is
matched (ie in the name of the MP)
# `( [A-Z]+)?` is used to match an Order of Australia award granted to the MP, awards are always
given as capitalised abbreviations
# and only 0 or 1 awards are given to each MP
# so we match `[A-Z]` (any capital letter), `+` repeats the previous expression 1 or more times
# then group that with a leading space and add a `?` to match 0 or 1 of the previous expression


4) Write a grep -E 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: grep -E 'y( [A-Z]+)?:.*y,' parliament.txt

# step 1: match a last name ending in y `y`
# step 2: match any Order of Australia award `( [A-Z]+)?`
# step 3: match the separator between the name and the electorate `:`
# step 4: match any number of characters in the electorate `.*`
# step 5: match the electorate ending in y `y`
# step 6: match the separator between the electorate and the state `,`


5) Write a grep -E 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: grep -E 'y( [A-Z]+)?:|y,' parliament.txt

# step 1: match a last name ending in y `y`
# step 2: match any Order of Australia award `( [A-Z]+)?`
# step 3: match the separator between the name and the electorate `:`
# OR `|`
# step 1: match the electorate ending in y `y`
# step 2: match the separator between the electorate and the state `,`


6) Write a grep -E command that will 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: grep -E 'ng\>' parliament.txt

# None of the Australian states or territories end in "ng" so we don't need to worry about matching
them
# `\b` matches a word boundary, that is the empty string immediately before or after a word
character
# `\<` is the same but only matches before a word character, `\>` is the same but only matches after
a word character


7) Write a grep -E 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: grep -E '[AEIOU]\S*[aeiou]( [A-Z]+)?:' parliament.txt

# We are once again using `:` to avoid matching any electorate (by placing it at the end of the
pattern)
# The first letter of a name should be capitalised so we can use `[AEIOU]` to match any capital
vowel
# The last letter of a name may or may not be capitalised so we can use `[aeiou]` to match any vowel
# We once again need to match any Order of Australia award `( [A-Z]+)?`
# Between the first and last letter we need to match any characters that are part of the name
# We cannot use `.*` because that would match spaces seperating the first and last name
# instead we use `\S` which matches any non-space character, this allows as to match double-
barrelled names like "Smith-Jones" or names like "O'Neil"
# while still disallowing spaces (or tabs or newlines)


8) Write a grep -E command that will print all the lines in the file where the electorate name
contains multiple words (separated by spaces or hyphens).
Q8 answer: grep -E ': Member for [a-zA-Z]+[ -][a-zA-Z]' parliament.txt

# We are once again using `:` to avoid matching any MP names (by placing it at the start of the
pattern)
# "Member for" is placed before every electorate name so we match that to get it out of the way
# `[a-zA-Z]+` matches a "word", that is 1 or more letters.
# `[ -]` matches either a space or a hyphen
# so we match a word, followed by a space or hyphen, followed by another word
```

EXERCISE:

Exploring Regular Expressions

You have been given a file named `ab_answers.txt` .
Which you must use to enter the answers for this exercise.

The autotest scripts depend on the format of `ab_answers.txt` .
So just add your answers where indicated but don't otherwise change the file.

```
# Open a text editor (gedit) in the background (&) and not owned by the current terminal (disown)
$ gedit ab_answers.txt & disown
# Or use any other text editor of your choosing
```

Use `grep -E` to test your answers to these questions.

We've provided a set of test cases in `input.txt`

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

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

ANSWER:

Sample answer:

```
$ grep -E 'A.*B|B.*A' input.txt
```

Once you have entered you answer you can check it like this:

```
$ 2041 autotest ab Q1
Test Q1 (ab Q1) - passed
1 tests passed 0 tests failed
```

2. Write a `grep -E` 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
A	BBAA
ABBA	ABBAA
ABAABAABAABBBBABB	ABBABABABABAA
ABAAAAAAAAAABBA	ABBBAAA
ABABABABA	BBABABABABABABAA

ANSWER:

Sample answer:

```
$ grep -E '^(BA|A)*(BA|B)*$' input.txt
```

Once you have entered you answer you can check it like this:

```
$ 2041 autotest ab Q2
Test Q2 (ab Q2) - passed
1 tests passed 0 tests failed
```

3. Write a `grep -E` 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	A
BABABABAB	AAAAA
AAAABBBBAAAA	ABABBBBBBBBBBBBBBBAAA
BBBAABBBBBAABBBAAAA	AAAABBABBAAAA
B	BBBAABBABBBAABBBAAAA

ANSWER:

Sample answer:

```
$ grep -E '^B*(AB*AB*AB*AB*)*$' input.txt
```

Once you have entered you answer you can check it like this:

```
$ 2041 autotest ab Q3
Test Q3 (ab Q3) - passed
1 tests passed 0 tests failed
```

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
```

before **Tuesday 13 June 12:00 (midday)** (2023-06-13 12:00:00) to obtain the marks for this lab exercise.

SOLUTION:

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 Q0 is: "grep -E Andrew words.txt"
Change the line that starts with
    "Q0 answer:"
to
    "Q0 answer: grep -E Andrew words.txt"
----------------------------------------------------------------------------------------------------
--------------------



1) Write a grep -E command that prints the lines in a file named input.txt containing at least one A
and at least one B.
Q1 answer: grep -E 'A.*B|B.*A' input.txt

# If we want one A and one B, we can wither have the A first followed by the B or the B first
followed by the A.
# So we create two patterns, one for each case, and use the | operator to combine them.
# The first pattern is A followed by anything followed by B.
# we don't care what is separating the A and B, so we use `.*` to match amount of any characters.


2) Write a grep -E 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.
Q2 answer: grep -E '^(BA|A)*(BA|B)*$' input.txt

# As we only want lines containing only A and B, there are only 4 possible combinations of adjacent
A's and B's.
# AA, AB, BA, BB
# we do not want the combination AA to occur after the combination BB.
# so we can only ever have AA, AB, or BA before BB
# This means that before the first BB, we can split the line into groups of A and BA
# as AA is just A, A and AB will either be part of A, BA, or be part of A, BB (from the second part
of the pattern)
# So the start of the line must be either BA or A repeated any number of times
# this gives us the first part of the pattern: `^(BA|A)*`
# the same logic is applied to the end of the line, but in reverse
# the second part of the line can be slit into repeated groups of BA and B
# this gives us the second part of the pattern: `(BA|B)*$`

3) Write a grep -E 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: grep -xE 'B*((AB*){4})*' input.txt

# `-x` is the same as adding `^` to the start of the pattern and `$` to the end of the pattern
# firstly we don't care about the number of B's, so we can just match any number of them until we
get to the first A: `B*`
# we then want to match a single A followed by any number of B's: `AB*`
# we then must have 4 of these groups: `(AB*){4}`
# we then can have 0 or more of these groups: `((AB*){4})*`
# that is 0 or more groups of 4 groups of A followed by any number of B's
# thus giving any multiple of 4 A's
```

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 **Week 3 Tuesday 12:00:00 (midday)** to submit your work.

You cannot obtain marks by e-mailing your code 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, using test cases different to those `autotest` runs for you. (Hint: do your own testing as well as running `autotest`.)

After automarking is run by the lecturer you can view your results here. The resulting mark will also be available 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
```