w3_regex

Regular Expression

grep (Global Regular Expression Print)

dictionary file `/usr/share/dict/words`

```
grep [-option] REGEX
```

options:

`-i` case-insensitive

`-v` invert match, that do not match the expression

`-E` extended regex

`-r` recursively search through directories

`-c` count lines

`-w` match whole word

`-o` only display the matched text and output each match in separate lines

REGEX:

`^` begin

`$` end

`[^ ]` exclude

`.` any character

`?` 0 or 1

`*` 0 or more

`+` 1 or more

`{n}` exactly n

`{m,n}` m or n

exercise

All words containing the letter capital Q.

```
    grep Q
```

All words starting with the letter R, in either upper or lower-case.

```
    grep -i R
```

All words ending in j.

```
    grep j$
```

The number of words containing the letter Q, ignoring case.

```
    grep -ic Q
    grep -i Q | wc -l
```

The first five words containing the letter sequence ‘cl’.

```
    grep cl | head -n 5
```

All words containing the sequence “kp”, but not “ckp”.

```
    grep [^c]kp
```

The last 15 words of exactly two letters.

```
grep ^..$ | tail -n 15
```

All three-letter words with no vowels (aeiou).

```
grep -iE ^[^aeiou]{3}$
```

All words of exactly 7 letters, where the third one is an e and the word ends “-ded”.

```
grep ^..e.ded$
```

Find all words that start with a P (whether capitalised or not), and contain at least four instances of the letter a.

```
grep ^[pP].*a.*a.*a.*a.*$
```

Contrive a file such that `grep` returns multiple lines but `grep -w` returns only one line.

```
echo -e "cat\ncatalog\ncater\nscatter" | grep -w 'cat'
```

find a situation where `grep -o PATTERNFILE | wc -l` and `grep -c PATTERNFILE` produce different results

```
echo -e "one cat two cat\none cat" | grep -c cat -> 2
echo -e "one cat two cat\none cat" | grep -o cat | wc -l -> 3
```

match both ‘encyclopaedia’ and ‘encyclopedia’ but nothing else.

```
grep -wE 'encyclopa?edia'
```

match UK postcodes.

```
grep -E [A-Z]{2}[0-9][\ ]?[0-9][A-Z]{2}
grep -E '[A-Z]{2}[0-9] ?[0-9][A-Z]{2}'
```

find an example that would match the following but fail to match the above.

```
^(
    ([A-Z]{1,2}[0-9][A-Z0-9]?|ASCN|STHL|TDCU|BBND|[BFS]IQQ|PCRN|TKCA) ?[0-9][A-Z]{2}
    |BFPO ?[0-9]{1,4}
    |(KY[0-9]|MSR|VG|AI)[ -]?[0-9]{4}
    |[A-Z]{2} ?[0-9]{2}
    |GE ?CX
    |GIR ?0A{2}
    |SAN ?TA1
)$
```

sed (Stream Editor)

```
sed [options] 's/SOURCE/DEST/'
```

options:

`-e` multiple commands in one line

`-E` extended regex

`'s/SOURCE/DEST/'`

`\( \)` create group in SOURCE to be referred in DEST

exercise

find all words ending in ‘ay’ and change ‘day’ into ‘week’.

```
grep ay$ | sed s/day/week/
```

In the same selection as above, replace all words that begin with ‘s’ with the word ‘sway’.

```
grep ay$ | sed s/^s/sway/
```

duplicate the match after a space, for any line containing ‘day’, so “saturday” becomes “saturday day”.

```
grep ay$ | sed 's/day/& &/'
```

any line ending in ‘day’ becomes a string “Xday or Xweek”, where X is the other part of the word.

```
grep ay$ | sed 's/\(.*\)day$/\1day or \1week/'
```

any word ending in either ‘way’ or ‘day’ to be flipped around and parenthesised, so ‘someday’ becomes ‘day (some)’ and ‘speedway’ becomes ‘way (speed)’.

```
grep ay$ | sed 's/\(.*\)\([dw]ay\)/\2 (\1)/'
```

difference between applying `s/a/e/` and `s/a/e/g`:

without g command only replace the first occurrence. `echo "banana" | sed 's/a/e/'` → “benana”.

with g command replace all occurrences. `echo "banana" | sed 's/a/e/g'` → “benene”.