Part Three

Substitution with sed

The power of sed

- delete specific lines or ranges of lines
- search and replace (with style)

sed syntax

```
sed [OPTIONS] <command>
sed [OPTIONS] '[LINE_ADDRESS] PROCEDURE'
```

sed workflow

for each line of input

    remove trailing newline character

    if line matches the address

        perform user's commands

    if **-n** option is not set

        append newline and print

sed will not hurt your data

sed reads your data and writes to STDOUT.

STDOUT will pour into your terminal unless redirected to a pipe or file.

Your original file is perfectly safe

sed won’t, but YOU can

NEVER REDIRECT TO ORIGIN

This will completely destroy your data. z.txt will be an empty file.

```
cat z.txt | prog1 | prog2 > z.txt # BAD!!!
```

Sample Data

Move into **`3rd`** folder, find the following:

```
m.tab - similar to unsorted.tab in Part 2
s.fa  - fasta file like those from Part 2
```

Addresses

1	Matches line number 1
2,5	Matches lines 2 to 5
5,$	Matches lines 5 and on
/to/	Matches lines with pattern ‘to’
1,/to/	Matches lines 1 to matching ‘to’
/b/,/a/	Matches from lines matching a to b

! operator, invert selection

Addresses can be negated with **!**

```
1!          Matches lines NOT equal to 1
2,5!        Matches lines NOT between 2 and 5
```

Procedure: deletion (d)

When the line matches the address, sed does not print, rather it moves onto the next line

Procedure: deletion (d)

The address can be a number or a regular expression:

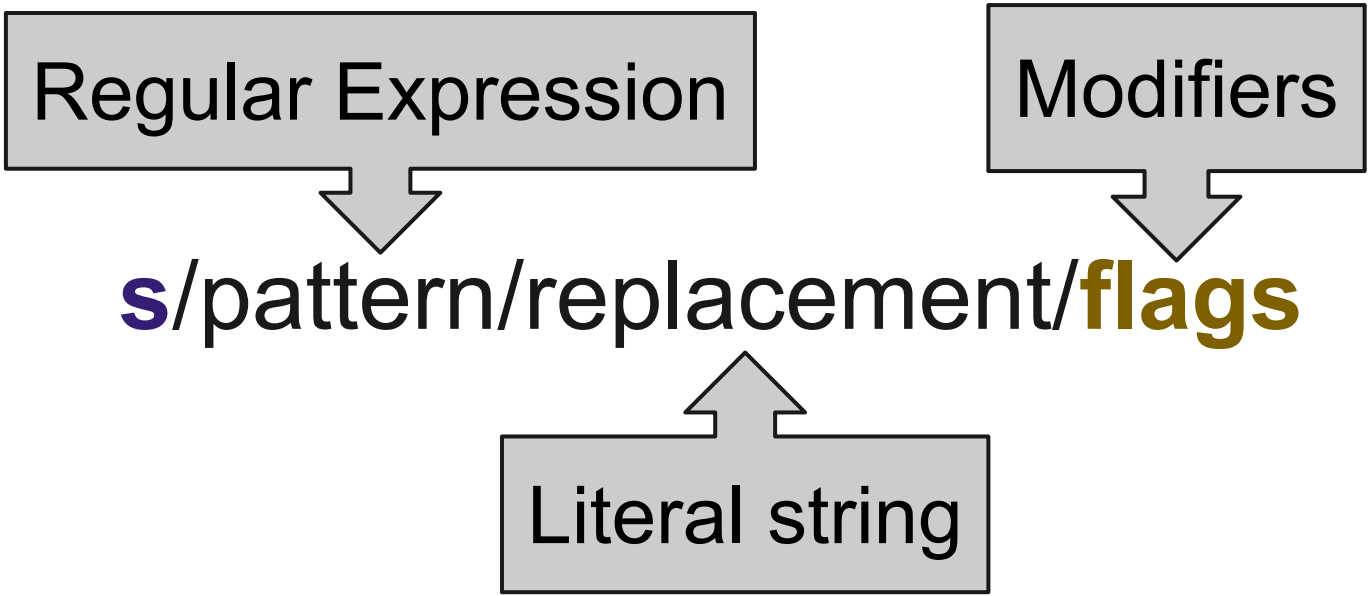
```
$ sed 'd' m.txt          # delete everything
$ sed '1d' m.txt         # delete 1st line
$ sed '/Fred/d' m.txt    # delete lines containing 'Fred'
$ sed '5,10d' m.txt      # delete lines 5 to 10
$ sed '10,$d' m.txt      # delete lines from 10 on
$ sed '/R/,/T/d' m.txt   # delete lines between R and T
$ sed '/R/,/T/!d' m.txt  # delete lines between R and T
```

DIY (1)

Spend a few minutes deleting lines in m.txt file

Try all the examples above

substitution (s)



Examples

```
# replace each line's 1st 'this' with 'that'
sed 's/this/that/'

# replace EVERY 'this' with 'that' (global flag)
sed 's/this/that/g'
```

Regular Expressions (1)

```
.       matches any character except a newline
*       matches 0 or more of the previous char
?       matches 0 or 1 of the precious char
[...]   matches any of the enclosed
[^...] matches everything EXCEPT the enclosed
^       anchors match at the BEGINNING of the line
$       anchors match at the END of the line
\       escapes the following special character
```

DIY(2)

```
$ sed ‘s/Fred/Franz/’ m.tab
$ sed ‘/Feb/,/Sep/ s/Bob/Larry/’ m.tab
$ sed ‘s/[1-5]/*/g’ m.tab
$ sed ‘s/\[.*\]//’ s.fa
$ sed ‘s/| .*//’ s.fa
```

Extended Expressions (2)

`(...)` captures the enclosed sequence

`\n` recalls *n*th captured sequence

`+` matches 1 or more of the previous characters

`|` OR

All of these require the **-r** argument (**-E** on mac)

Examples

```
# replace entire line with captured integer
$ sed -r ‘s/^>gi\|([0-9]+).*/\1/’ s.fa
# remove leading and trailing space
$ sed -r ‘s/^ +| +$//g’
# substit
$ sed -r ‘/Jan/,/Jul/ s/Olga|Fred/Oleg/g’
```

Print only if substituted

Problem:

```
$ sed -r 's/^>gi\|([0-9]+).*/\1/' s.fa
```

You want is a list of integers, but all the lines in the input still print

Solution:

```
$ sed -rn 's/^>gi\|([0-9]+).*/\1/p' s.fa
```

p flag and -n option

The **p** flag in substitution forces the line to be printed

The **-n** option suppresses printing by default

Therefore, print only if substitution succeeds

Extraction strategy

To one of more words from a line:

- Start with the term to be extracted

```
sed -rn 's/.*([0-9]+).*/\1/p'
```

- Make pattern unambiguous by adding context

```
sed -r 's/.*id=([0-9]+).*/\1/'
```

- If no context is necessary, just use `grep -o`

```
grep -oE '[0-9]+'
```

DIY (3)

s.fa headers are basically formatted as so:

```
>gi|<number>|ref|<string>| <description> [<species>]
```

Try to extract these regions as single columns