Assignment 2: Eddy

version: 0.2 last updated: 2024-03-07 11:00

Aims

This assignment aims to give you:

* practice in Python programming generally.
* a clear and concrete understanding of sed's core semantics.

Introduction

Your task in this assignment is to write a Python program eddy.py which implement the ***Eddy*** editing commands described below.

*Eddy* editing commands are a simple subset of the important tool [Sed](https://en.wikipedia.org/wiki/Sed) which you met earlier in the course.

[Sed](https://en.wikipedia.org/wiki/Sed) is a *very* complex program that has many commands.  
***Eddy*** contains only a few of the most important [Sed](https://en.wikipedia.org/wiki/Sed) commands.  
There are also some simplifying assumptions below, which make your task easier.

You must implement **Eddy** in Python only. The **Permitted Languages** section below has more information.

The more challenging parts of this assignment may require some research.  
You may need to explore documentation and other information for Python, Sed, and regexes.  
Searching for this type of information is a *very* useful skill to practice.,

Reference implementation

Many aspects of this assignment are not fully specified in this document;  
instead, you must match the behaviour of the reference implementation: **2041 eddy**

Provision of a reference implementation is a common method to provide or define an operational specification,  
and it's something you will likely need to do after you leave UNSW.

Discovering and matching the reference implementation's behaviour is deliberately part of the assignment,  
and will take some thought.

If you discover what you believe to be a bug in the reference implementation, report it in the class forum.  
Andrew and Dylan may fix the bug, or indicate that you do not need to match the reference implementation's behaviour in this case.

Eddy Commands

Subset 0

In subset 0 eddy.py will always be given a single Eddy command as a command-line argument.

The Eddy command will be one of 'q', 'p', 'd', or 's' (see below).

The only other command-line argument possible in subset 0 is the **-n** option.

Input files will not be specified in subset 0.

For subset 0 eddy.py need only read from standard input.

Subset 0: q - quit command

The Eddy **q** command causes eddy.py to exit, for example:

**seq 1 5 | 2041 eddy '3q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 9 20 | 2041 eddy '3q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 10 15 | 2041 eddy '/.1/q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 500 600 | 2041 eddy '/^.+5$/q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 100 1000 | 2041 eddy '/1{3}/q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

*Eddy* commands are applied to input lines as they are read.

The **q** command means eddy.py may not read all input.

For example, the command prints an "infinite" number of lines containing (by default) "yes".

**yes | 2041 eddy '3q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

This means eddy.py can not read all input first, e.g. into a list, before applying commands.

Subset 0: p - print command

The Eddy **p** commands prints the input line, for example:

**seq 1 5 | 2041 eddy '2p'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 7 11 | 2041 eddy '4p'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 65 85 | 2041 eddy '/^7/p'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 5 | 2041 eddy 'p'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 0: d - delete command

The Eddy **d** command deletes the input line, for example:

**seq 1 5 | 2041 eddy '4d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 100 | 2041 eddy '/.{2}/d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 11 20 | 2041 eddy '/[2468]/d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 0: s - substitute command

The Eddy **s** command replaces the specified regex on the input line.

**seq 1 5 | 2041 eddy 's/[15]/zzz/'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 10 20 | 2041 eddy 's/[15]/zzz/'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 100 111 | 2041 eddy 's/11/zzz/'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

The substitute command can be followed optionally by the modifier character **g**, for example:

**echo Hello Andrew | 2041 eddy 's/e//'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**echo Hello Andrew | 2041 eddy 's/e//g'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**g** is the only permitted modifier character.

Like the other commands, the substitute command can be given addresses to be applied to:

**seq 11 19 | 2041 eddy '5s/1/2/'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 51 60 | 2041 eddy '5s/5/9/g'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 100 111 | 2041 eddy '/1.1/s/1/-/g'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 0: -n command line option

The Eddy **-n** command line option stops input lines being printed by default.

**seq 1 5 | 2041 eddy -n '3p'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 2 3 20 | 2041 eddy -n '/^1/p'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**-n** command line option is the only useful in conjunction with the **p** command,  
but can still be used with the other commands.

Subset 0: Addresses

All Eddy commands in subset0 can optionally be preceded by an address specifying the line(s) they apply to.

In subset 0, this address can either be a line number or a regex.

The line number must be a positive integer.

The regex must be delimited with slash **/** characters.

Subset 0: Regexes

In subset 0, you can assume backslashes **\** do **not** appear in address or substitution regexes.

In subset 0, you can assume semicolons **;** do **not** appear in address or substitution regexes.

In subset 0, you can assume commas **,** do **not** appear in address or substitution regexes.

In subset 0, regexes are delimited with slash **/** characters, so you can assume slashes do not appear in regexes.

In subset 0 and all other subsets, you can assume the regex is correct. You do not have to check for errors in the regex.

In subset 0 and all other subsets, you can assume the regex is a POSIX-compatible extended regular expression.

In subset 0 and all other subsets, you can assume the regex is compatible with Python.  
In other words, the regex can be used directly as a Python regular expression, for example passed to re.search, and will have the same meaning.

Note, if testing regular expressions with sed, you need to specify sed -E for extended regular expressions to work.

Subset 1

Subset 1 is more difficult. You will need to spend some time understanding the semantics (meaning) of these operations, by running the reference implementation and researching the equivalent **sed** operations.

Note the assessment scheme recognises this difficulty.

Subset 1: s - substitute command

In subset 1, any non-whitespace character may be used to delimit a substitute command, for example:

**seq 1 5 | 2041 eddy 'sX[15]XzzzX'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 5 | 2041 eddy 's?[15]?zzz?'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 5 | 2041 eddy 's\_[15]\_zzz\_'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 5 | 2041 eddy 'sX[15]Xz/z/zX'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 1: Multiple Commands

In subset 1, multiple Eddy commands can be supplied separated by semicolons **;** or newlines. For example:

**seq 1 5 | 2041 eddy '4q;/2/d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 5 | 2041 eddy '/2/d;4q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 20 | 2041 eddy '/2$/,/8$/d;4,6p'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 5 | 2041 eddy '4q**

**/2/d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 5 | 2041 eddy '/2/d**

**4q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Semicolons can not appear elsewhere in subset 1 commands.

Subset 1: -f command line option

The Eddy **-f** reads Eddy commands from the specified file, for example:

**echo 4q > commands.eddy**

**echo /2/d >> commands.eddy**

**seq 1 5 | 2041 eddy -f commands.eddy**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**echo /2/d > commands.eddy**

**echo 4q >> commands.eddy**

**seq 1 5 | 2041 eddy -f commands.eddy**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

commands can be supplied separated by semicolons **;** or newlines.

Subset 1: Input Files

In subset 1, input files can be specified on the command line:

**seq 1 2 > two.txt**

**seq 1 5 > five.txt**

**2041 eddy '4q;/2/d' two.txt five.txt**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 2 > two.txt**

**seq 1 5 > five.txt**

**2041 eddy '4q;/2/d' five.txt two.txt**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**echo 4q > commands.eddy**

**echo /2/d >> commands.eddy**

**seq 1 2 > two.txt**

**seq 1 5 > five.txt**

**2041 eddy -f commands.eddy two.txt five.txt**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 1: Comments & White Space

In subset 1, whitespace can appear before and/or after commands and addresses.

In subset 1, '#' can be used as a comment character, for example:

**seq 24 43 | 2041 eddy ' 3, 17 d # comment'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

On both the command line and in a command file, a newline ends a comment

**seq 24 43 | 2041 eddy '/2/d # delete ; 4 q # quit'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 1: Addresses

In subset 1, **$** can be used as an address.  
It matches the last line, for example:

**seq 1 5 | 2041 eddy '$d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 1 10000 | 2041 eddy -n '$p'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Eddy can read one line of input ahead to handle **$** addresses.

In subset 1, Eddy commands can optionally be preceded by a comma-separated pair of addresses specifying the start and finish of the range of lines the command applies to, for example:

**seq 10 21 | 2041 eddy '3,5d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 10 21 | 2041 eddy '3,/2/d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 10 21 | 2041 eddy '/2/,4d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 10 21 | 2041 eddy '/1$/,/^2/d'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**seq 10 30 | 2041 eddy '/4/,/6/s/[12]/9/'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Comma-separated pairs of addresses can not be used with the q command.

Subset 1: Regexes

All the rules from Subset 0 about regexes still apply, except:

In subset 1, substitute regexes are **not always** delimited with slash **/** characters,  
So you can **not** assume slashes do not appear in regexes.  
You **can** assume that whatever the delimiter is, it will not appear in the substitute regex.  
Only substitute regexes can be delimited with other characters, address regex are always delimited by slashes.

Subset 2

Subset 2 is even more difficult. You will need to spend considerable time understanding the semantics of these operations, by running the reference implementation, and/or researching the equivalent **sed** operations.

Note the assessment scheme recognises this difficulty.

Subset 2: s - substitute command

In subset 2, any character, including the character used to delimit the substitute command, may appear in the regex or replacement string.

In subset 2, backslash may appear in the regex or replacement string.

Subset 2: -i command line option

The Eddy **-i** command line option replaces file contents with the output of the Eddy commands. You should use a temporary file.

**seq 1 5 > five.txt**

**cat five.txt**

1

2

3

4

5

**2041 eddy -i /[24]/d five.txt**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**cat five.txt**

1

2

3

4

5

Subset 2: Multiple Commands

In subset 2, semicolons **;** and commas **,** can appear inside Eddy commands.

**echo 'Punctuation characters include . , ; :' | 2041 eddy 's/;/semicolon/g;/;/q'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 2: : - label command

The Eddy **:** command indicates where **b** and **t** commands should continue execution.

There can not be an address before a label command.

Subset 2: b - branch command

The Eddy **b** command branches to the specified label, if the label is omitted, it branches to the end of the script.

Subset 2: t - conditional branch command

The Eddy **t** command behaves the same as the **b** command except it branches only if there has been a successful substitute command since the last input line was read and since the last **t** command.

**echo 1000001 | 2041 eddy ': start; s/00/0/; t start'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

**echo 0123456789 | 2041 eddy -n 'p; : begin;s/[^ ](.)/ \1/; t skip; q; : skip; p; b begin'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 2: a - append command

The Eddy **a** command appends the specified text.

**seq 5 9 | 2041 eddy '3a hello'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 2: i - insert command

The Eddy **i** command inserts the specified text.

**seq 5 9 | 2041 eddy '3i hello'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

Subset 2: c - change command

**seq 5 9 | 2041 eddy '3c hello'**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

The Eddy **c** command replaces the selected lines with the specified text.

Subset 2 Assmptions: Regexes

In subset 2, backslash **\** may appear in regexes.

In subset 2, any character including the character used to delimit the regex may appear in the regex itself.

Other Sed Features

You do not have to implement in Eddy sed features and commands other than those described above.

For example, sed on CSE systems provides extra commands including **{} D h H g G l n p T w W x y** which are not part of Eddy.

For example, sed on CSE systems adds extra syntax to addresses including features involving the characters: **! + ~ 0 \**. These are not part of Eddy.

For example, sed on CSE systems has a number of command-line options other than -i, -n and -f. These are not part of Eddy

The reference implementation implements many of these extra sed features and commands.

The marking will not test your code on these extra features and commands.

You do not have to check for these extra features and commands.

You will not be penalized if you choose to implement any of these extra features and commands.

Assumptions/Clarifications - All Subsets

Like all good programmers, you should make as few assumptions as possible.

You can assume that only the arguments described above are supplied to eddy.py. You do not have to handle other arguments.

You must apply the Eddy commands to input lines as you read the input lines. You can not read all input lines first (e.g. into a list). There may be an unlimited number of input lines.

You are permitted to read one line ahead to handle **$** addresses.

You are permitted to read one line ahead even if the commands do not use a **$** address.

You should match the output streams used by the reference implementations. It writes error messages to stderr: so should you.

You should match the exit status used by the reference implementation. It exits with status 1 after an error: so should you.

You can assume arguments will be in the position and order shown in the usage message from the reference implementation. Other orders and positions will not be tested. Here is the usage message:

**2041 eddy**

/usr/local/bin/2041: line 122: /web/cs2041/bin/eddy: cannot execute: required file not found

You can assume, Eddy regular expressions are valid Python regular expressions and are compatible with Python. In other words, they can be used as Python regular expressions and will have the same effect.

You can assume command line arguments, STDIN and all files contain only ASCII bytes.

You can assume all input lines in STDIN and in all files are terminated by a '\n' byte.

Eddy error messages include the program name. It is recommended you use sys.argv[0] however it is also acceptable to hard-code the program name. The automarking and style marking will accept both.

Testing