Part I: Regexes

Submission: on Canvas, under Assignments
Deadline: week 5, Thurs 16th of Feb 2023

Notes on answering this question: In my experience the most convincing presentation for your answers is to highlight the non-matching regexes in RED and the matching regexes in GREEN. So you end up with a pretty multicolour list of QandAs, and the marker just scans the page looking for the right pattern.

For questions 1-9, write a description of what the regex pattern describes in plain English.

In the questions below, “matches” refers to whole-string matching (and not to matching a substring).

1. Which of the following matches regex /abababa/?
   1) abababa
   2) aaba
   3) aabbaa
   4) aba
   5) aabababa

The expression matches the same string which is abababa.

2. Which of the following matches regex /ab+c?/?
   1) abc
   2) ac
   3) abbb
   4) bbc

This expression expects “a” and then one or more “b” characters and then c at last.

3. Which of the following matches regex /`a.[bc]+`/?
   1) `abc`
   2) `abbbbbbbb`
   3) `azc`
   4) `abcbcbcbc`
   5) `ac`
   6) `asccbbbbcbcccc`

This expression expects “a” and then any character with “bc” one or more time.

4. Which of the following matches regex /`abc|xyz`/?
   1) `abc`
   2) `xyz`
   3) `abc|xyz`

This expression expects abc or xyz.

5. Which of the following matches regex /`[a-z]+[\.\?!]`/?
   1) `battle!`
   2) `Hot`
   3) `green`
   4) `swamping.`
   5) `jump up.`
   6) `undulate?`
   7) `is.?`

This expression expects one or more small alphabets with “.” or “?” or ! at last.

6. Which of the following matches regex /`[a-zA-Z]*[^,]=`/?
   1) `Butt=`
   2) `BotHEr,=`
   3) `Ample`
   4) `FIdDlE7h=`
   5) `Brittle =`
   6) `Other.=`

This expression expects zero or more alphabets and then it expects any character other than “,” and then expects “=”.

7. Which of the following matches regex /`[a-z][\.\?!]\s+[A-Z]`/?
(\s matches any space character)
   1) `A. B`
   2) `c! d`
   3) `e f`
   4) `g. H`
   5) `i? J`
   6) `k L`

   This expression expects a single small case alphabet with “.” or “?” or “!” and then multiple space characters followed by a single capital alphabet.
8. Which of the following matches regex /`(very )+(fat )?(tall|ugly) man`/?
   1) `very fat man`
   2) `fat tall man`
   3) `very very fat ugly man`
   4) `very very very tall man`

   This expression expects one or more “very “ with zero or one “fat “ followed by a word tall or ugly and then man.

9. Which of the following matches regex /`<[^>]+>`/?
   1) `<an xml tag>`
   2) `<opentag> <closetag>`
   3) `</closetag>`
   4) `<>`
   5) `<with attribute=”77”>`

This expressions expects first character to be “<” and then one or more characters other than “>” and then after multiple characters it has to be “>”.

10. Write a regex to identify dates of the form dd/mm/yyyy.
    I expect dd to range from 01 to 31, and mm to range from 01 to 12, and I expect yyyy to range from 0001 to 9999 and in particular to *not* be 0000 (the Gregorian calendar predates this; see Year 0 and the invention of 0). However, I do not expect you to cross-reference mm against dd or to restrict yyyy, so that e.g. 31/02/0231 is fine.
    Do **not** use backreferences or negative lookahead (so if your answer contains `?!`, then it’s not admissible for this question).

(0[1-9]|[12][0-9]|3[01])\/(0[1-9]|[1][012])\/(0{3}[1-9]|0{2}[1-9][0-9]|0[1-9][0-9]{2}|[1-9][0-9]{3})

11. Write a regex to identify dates of the form dd/mm/yyyy or dd.mm.yyyy, but *not* using mixed separators such as dd/mm.yyyy. You may use backreferences, negative lookahead, and other fancy tricks, if convenient.

```
(
((0[1-9]|[12][0-9]|3[01])\/(0[1-9]|[1][012])\/(0{3}[1-9]|0{2}[1-9][0-9]|0[1-9][0-9]{2}|[1-9][0-9]{3}))
|
((0[1-9]|[12][0-9]|3[01])\.(0[1-9]|[1][012])\.(0{3}[1-9]|0{2}[1-9][0-9]|0[1-9][0-9]{2}|[1-9][0-9]{3}))
|
((0[1-9]|[12][0-9]|3[01])-(0[1-9]|[1][012])-(0{3}[1-9]|0{2}[1-9][0-9]|0[1-9][0-9]{2}|[1-9][0-9]{3}))
)
```

12. *(Unmarked)* Explain whether a regex can identify correct bracketing, as in `((a))` but not `(((a)`.

Yes, the regex can identify the string.

13. *(Unmarked)* Explain the relevance of the regex /`Whiske?y`/

The regex /Whiske?y/ checks two types of words. The words are Whiskey and Whisky. The question mark(?) in the expression checks whether the letter “e” is included zero or one time. If the letter is included 0 or 1 time the result matches to the expression. If the letter is included more than one time then it does not matches the expression.

14. Write a regex for the set of even numbers without leading zeroes (base 10; so the alphabet is `[0-9]`). Note that 0 and 2 and 10 and 20 are even numbers without leading zeroes, and 00 and 02 and 1 and 01 are not even numbers without leading zeroes. Check that your regex accepts 100 and 10012.

```
([1-9][0-9]*[02468]$|[02468])
```