Horsefez's Guide to Regex Mastery

About the author

Matthias Kowalewski, the author of this document, is, despite popular belief, not a real horse. He loves regex as much as being random.

If he is not actively hosting regex challenges in the Splunk> community he works as a Splunk> Consultant with focus on IT-Security. He also enjoys playing strategy games on his computer or hacking into vulnerable machines that were set up in his private security lab.

Find Matthias on LinkedIn: www.linkedin.com/in/matthiaskowalewski

Licensing

In the words of the author

Use this guide to challenge your friends or colleagues. Share it with everyone.
Be aware that it is prohibited to use the presented material in any commercial way. You are not allowed to make money with the content of this document. Do not use the content of this document elsewhere without appropriately crediting the author and origin.

Glossary

Introduction

About this document

This document serves as a collection of regular expression (regex) challenges held by **horsefez** during the 'Regex Tuesday' events in the Splunk> community user group on slack. This document includes ten challenges, plus one bonus challenge that has never been revealed before.

How to use this document?

This document should encourage you to overcome your struggles with regex by challenging you in fun and creative ways. Challenge yourself, your friends or coworkers to a regex-off and find out who the real regex master is.

What regex flavor should I use?

You can use any regular expression flavor you like or feel comfortable with. Just note that the challenges were originally run using PCRE (PHP<7.3). Scores were calculated with the PCRE engine implemented on regex101.com.

What regex editor should I use?

There are multiple regular expression editors and engines on the web. The author prefers to use regex101.com as it is easy to use and comes with a debugger option, quick-reference, explanations and a mode to use substitution. It additionally allows you to save, fork and share your solutions with friends and colleagues.

Where can I learn more about regular expressions?

Like with regex editors there are also multiple learning options when it comes to regular expressions. Sites like rexegg.com and regular-expressions.info are good starting points. These are the sites where the author learned and improved his skills. There is also a fun site called regexcrossword.com where you are able to fill out crossword-puzzles using text that has to match regular expression statements.
There are also so called 'regex-golf' sites, but the author discourages you from trying them out as a beginner, as a lot of those so called 'challenges' require you to use brute-force to get to the solution.

About the challenges

The challenges were all already tested in the field. Meaning that countless people have tried them already and came up with solutions - therefore are doable. They might look difficult or impossible to some of you at first glance, but you can trust the author that they are indeed possible.

How difficult are the challenges?

The author tried to sort the challenges in a way so it starts fairly easy and then ramps up the difficulty in the later ones. This document uses a horse-based difficulty ranking system (HbDRS) which shows the difficulty using horse figures. Example of HbDRS in action:

Difficulty Rating “Easy”	=	🐴
Difficulty Rating “Medium”	=	🐴🐴
Difficulty Rating “Hard”	=	🐴🐴🐴
Difficulty Rating “Insane”	=	🐴🐴🐴🐴🐴

Did horsefez come up with all the challenges by himself?

No, not at all. Many of the challenges present in this collection were found on the web. Big thanks go out to **Callum Macrae** who not only gave me the idea for the ‘Regex Tuesday’ events but also provided a lot of the challenges. Fun fact, they aren’t his either as he also just found them on the web and collected them on his website here callumacrae.github.io/regex-tuesday/.

The challenges on Callum Macrae’s site were designed to be done by using the JavaScript implementation of regex. The author went ahead and made them PCRE compatible, added additional descriptions, wrote an extended ruleset and adjusted the challenge data.

Feedback

I would love to hear your feedback on all things regex.
Check out the Contact me for feedback! section for information on how to reach me.

Introduction to the Challenges

Structure of challenges

Title

The title of the challenge.

Difficulty (HbDRS)

Difficulty assumption done by the author.

Preamble

An optional short introduction story.

Description

A description on what to actually achieve in the challenge.

Rules

The ruleset for the current challenge. Mostly the same on all challenges.

Winning Categories

The categories that were judged in the original installment of the challenge.
Mostly ‘fewest steps’ and ‘fewest chars’. Be aware that most of the time there is one solution that has an optimal step count, but not an optimal char count and vice versa. Solutions that have good scores in both categories are rare.

Score-Self-Check

Shows score ranges, so you can better evaluate your own solutions.

Hints

Optional hints the author might have.

Challenge Data

The data you need to match. Put it into the ‘test string’ section on regex101.

Expected Output

What the end results should look like.

Helpful Links

Links to techniques that may help you in the challenge.

Additional info

Default Rules

Do not try skipping over unwanted words/lines by using tricks like this or similar:
(?<match1>\w{15})\s*\d{7}\n(?<match2>\w{36})
Instead write logic that works with the content of every line independently.

Do not use anchors like \A, \Z, \z or \G when writing your regular expression.
Instead use anchors like \b, \B, ^ or $.

Do not make your regex fail on purpose when it encounters the part of data that should not match using trickery with regex control-verbs.
Instead write a regex that evaluates every line of the data individually. Regex control-verbs are allowed to be used. But don't try to get them to work as a beginner.

Do not use flags 'A' and 'J'.
Use flags 'g' (global) and 'm' (multiline) as default. Other flags are also allowed.

Debugger

On regex101.com there is a neat debugger built into the site. You can find it on the left side of the screen and should use it to optimize your step count after you've come up with a working solution or when you need to understand how the regex engine actually works.

Yes, I encourage you to make use of this feature extensively.

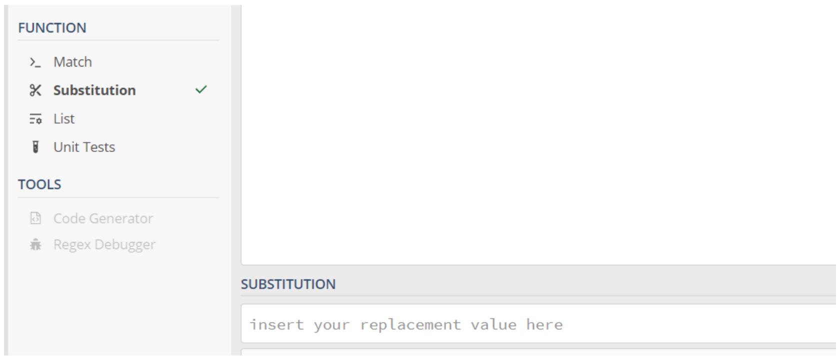
Stuck somewhere?

Oftentimes it is useful to start from scratch. If you struggle to solve a problem in life it is often good to look at it from another angle. Regex is no different. Save what you have and start anew.

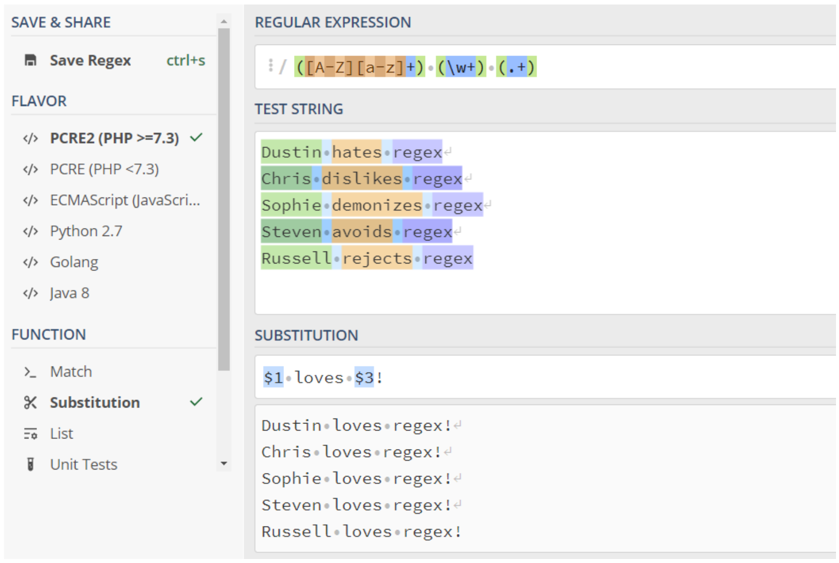
Substitution function

Some of the upcoming challenges require you to use the built-in substitution function of regex101.com. It might frighten you a bit at first, but let me assure you that it is rather easy to use.

You can find the substitution function here. Just click on it and it shows up.



Let us make a short example to show you how it works and what it actually does.



```
/ ([A-Z][a-z]+) (\w+) (.+)
```

```
Dustin hates regex
Chris dislikes regex
Sophie demonizes regex
Steven avoids regex
Russell rejects regex
```

```
$1 loves $3!
```

```
Dustin loves regex!
Chris loves regex!
Sophie loves regex!
Steven loves regex!
Russell loves regex!
```

With the power of regex, we changed their opinion about regular expressions. Neat.

You can find another good example and explanation over here:
https://sodocumentation.net/regex/topic/9852/substitutions-with-regular-expressions

The Challenges

The famous writer

HbDRS

Preamble

In a land far far away...

A famous novel writer once told me the biggest challenge he was facing whenever he had finished a new book was correcting mistakes he made while writing it.
One of the problems was that whenever he drank alcoholic beverages before the writing session he would occasionally repeat words twice 'twice'.
It previously was very difficult for him to iron out those mistakes whenever he did some proofreading afterwards. Luckily for him, he employed you to solve his problems.

Description

Find the words that are occuring twice twice right after another. Remember that those words could also be case CASE sensitive.
It can happen that some words contain other words in them and therefore will also match if the regex is written poorly. Only words that are coming right after each other should be matched.
To help the novel author to better find the words he should delete we are going to encase the second word in html tags, formatting it to **bold text** using <b>bold text</b>.

You need to use the built-in 'substitution' function of regex101.

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex + Count of characters of the substitution

Score-Self-Check

Fewest steps:

Good ~ 1000 steps; Great ~ 600 steps; Amazing ~ 450 steps

Fewest chars:

Good ~ 50 chars; Great ~ 40 chars; Amazing ~ 30 chars

Hints

(\b)	Word boundaries.
(\1)	Backreference.
(?i)	Flags.

Challenge Data

```
This is a text
This is is a text
This text text is is
This text is a text
This test text is a test
This this text is a text
cat dog dog cat dog
This test is a test tester
hello world hello world
This nottest test is something
This is IS a test
<Westy> I'll I'll be be back back soon soon.
```

Expected Output (10 matches)

```
This is a text
This is <b>is</b> a text
This text <b>text</b> is <b>is</b>
This text is a text
This test text is a test
This <b>this</b> text is a text
cat dog <b>dog</b> cat dog
This test is a test tester
hello world hello world
This nottest test is something
This is <b>IS</b> a test
<Westy> I'll <b>I'll</b> be <b>be</b> back
<b>back</b> soon <b>soon</b>.
```

Helpful Links

https://www.regular-expressions.info/wordboundaries.html

https://www.regular-expressions.info/backref.html

https://www.regular-expressions.info/modifiers.html

The end of your sentence

HbDRS

Preamble

After you have successfully helped out our novel writer from the first challenge he contacts you a while later with a new request. He wants you to match sentences in pairs of two out of his latest script. The reason behind this request is unknown, but he pays you good money so you won't ask further questions.

Description

You need to separate sentences at the natural sentence-break-points (made up word) by splitting them into two separate matches using named capture groups:

- first_sentence
- second_sentence

example:	*Joseph Hornsby works for Splunk. I am his biggest fan and*

regex:	*(?<first_sentence>some regex logic) (?<second_sentence>some other logic)*
first_sentence:	*Joseph Hornsby works for Splunk.*
second_sentence:	*I am his biggest fan and*

Do not capture the space between the sentences. In most cases it is just one whitespace.
Do use named capture groups to capture both sentences individually.

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex

Score-Self-Check

Fewest steps:

Good ~ 1000 steps; Great ~ 300 steps; Amazing ~ 105 steps

Fewest chars:

Good ~ 90 chars; Great ~ 75 chars; Amazing ~ 65 chars

Hints

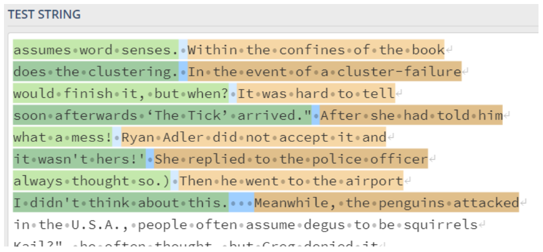
([...] or [^...])   Character classes.

Challenge Data

```
assumes word senses. Within the confines of the book
does the clustering. In the event of a cluster-failure
would finish it, but when? It was hard to tell
soon afterwards ‘The Tick’ arrived." After she had told him
what a mess! Ryan Adler did not accept it and
it wasn't hers!' She replied to the police officer
always thought so.) Then he went to the airport
I didn't think about this.   Meanwhile, the penguins attacked
in the U.S.A., people often assume degus to be squirrels
Kail?", he often thought, but Greg denied it
the goose weighed 13.5 kilograms, which was a lot
well... they'd better not install that software
J.H. has long been a very talented Operations Analyst at Splunk
like that", James M. thought, but was pleasantly surprised when
but W. G. Grace never had much hope in Thomas Turner
```

Expected Output (8 matches)

The first eight (8) sentences from the top downwards should match using your regex logic. The rest of the sentences should not match using your logic.


TEST STRING

```
assumes word senses. Within the confines of the book
does the clustering. In the event of a cluster-failure
would finish it, but when? It was hard to tell
soon afterwards ‘The Tick’ arrived." After she had told him
what a mess! Ryan Adler did not accept it and
it wasn't hers!' She replied to the police officer
always thought so.) Then he went to the airport
I didn't think about this.   Meanwhile, the penguins attacked
in the U.S.A., people often assume degus to be squirrels
Kail?", he often thought, but Greg denied it
```

Helpful Links

https://www.regular-expressions.info/charclass.html

https://www.regular-expressions.info/named.html

The 29th of February

HbDRS

Preamble

Let’s just assume that the novel author is an immortal being and wants to publish his books only on the 29th of February of every leap-year. He asks you to match only the 29th of February’s that are valid in a list of dates he provides, so that he knows when he has to publish new literature. Also match dates that are already in the past.

And so ends the tale of the famous writer who was able to solve all his issues with the power of regex and he then lived happily ever after. Or does he? We’ll see.

Description

It is kind of obvious that regular expression is not able to model a complex logic such as validating which 29th of February is legit or not. But it is certainly possible to match only the correct dates using a brute-force approach.

Check the ‘Expected Output’ section for further clarification.

Rules

Do not use anchors like \A, \Z, \z or \G. **Do not** use flags ‘A’ and ‘J’. Everything else is allowed.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex

Score-Self-Check

Fewest steps:
Good ~ 2000 steps; Great ~ 1300 steps; Amazing ~ 700 steps
Fewest chars:
Good ~ 50 chars; Great ~ 43 chars; Amazing ~ 35 chars

Hints

Character classes. Negated Character classes. Repetition.

Challenge Data

29th of February 1998
29th of February 1999
29th of February 2000
29th of February 2001
29th of February 2002
29th of February 2003
29th of February 2004
29th of February 2005
29th of February 2006
29th of February 2007
29th of February 2008
29th of February 2009
29th of February 2010
29th of February 2011
29th of February 2012
29th of February 2013
29th of February 2014
29th of February 2015
29th of February 2016
29th of February 2017
29th of February 2018
29th of February 2019
29th of February 2020
29th of February 2021
29th of February 2022
29th of February 2023
29th of February 2024
29th of February 2025
29th of February 2026
29th of February 2027
29th of February 2028
29th of February 2029
29th of February 2030
29th of February 2031
29th of February 2032
29th of February 2033
29th of February 2034
29th of February 2035
29th of February 2036
29th of February 2037
29th of February 2038
29th of February 2039
29th of February 2040
29th of February 2041
29th of February 2042
29th of February 2043
29th of February 2044
29th of February 2045
29th of February 2046
29th of February 2047
29th of February 2048
29th of February 2049
29th of February 2050
29th of February 2051
29th of February 2052
29th of February 2053
29th of February 2054
29th of February 2055
29th of February 2056
29th of February 2057
29th of February 2058
29th of February 2059
29th of February 2060
29th of February 2061
29th of February 2062
29th of February 2063
29th of February 2064
29th of February 2065
29th of February 2066
29th of February 2067
29th of February 2068
29th of February 2069
29th of February 2070
29th of February 2071
29th of February 2072
29th of February 2073
29th of February 2074
29th of February 2075
29th of February 2076
29th of February 2077
29th of February 2078
29th of February 2079
29th of February 2080
29th of February 2081
29th of February 2082
29th of February 2083
29th of February 2084
29th of February 2085
29th of February 2086
29th of February 2087
29th of February 2088
29th of February 2089
29th of February 2090
29th of February 2091
29th of February 2092
29th of February 2093
29th of February 2094
29th of February 2095
29th of February 2096
29th of February 2097
29th of February 2098
29th of February 2099
29th of February 2100
29th of February 3066
29th of February 4040
29th of February 7072
29th of February 8022
29th of February 9996

Expected Output (28 matches)

There are 28 valid dates. 28 lines should match your logic. The rest of the lines should not match your logic.

It is a brute-force approach. It might get ugly, but it doesn't have to. If you get stuck anywhere remember that starting from scratch can oftentimes help.

Helpful Links

https://www.regular-expressions.info/charclass.html
https://www.regular-expressions.info/charclasssubtract.html
https://www.regular-expressions.info/repeat.html
https://www.timeanddate.com/date/leapyear.html

Numbers

HbDRS 🐴🐴

Preamble

This challenge marks the start of the 'medium' difficulty challenges. By now you should have a basic understanding of regular expressions.

Description

In this challenge you need to validate certain number formats. Write an expression that would also work with different numbers in the same formats. Do not brute-force.

Your matching numbers need to be captured in a named capturing group called "match".

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex

Score-Self-Check

Fewest steps:
Good ~ 1500 steps; Great ~ 1100 steps; Amazing ~ 900 steps
Fewest chars:
Good ~ 100 chars; Great ~ 75 chars; Amazing ~ 60 chars

Hints

Possessive Quantifiers. Optional Items. Alternation.

Challenge Data (match, no match)

124
1,024
2,000,204
3,000.6
8,205,500.4672
0.5
36,000.57
100,000
5
42
10,5
10.5
10.44444444
1 024
9 999 352
10,19836
30 000,7302
0,5
47 372
10,000,000.45
10 000 000,45
123,456,789
123 456 789
1,05335
1.53252
.5
1025
1,1337,000
,046
100.
2.2.2
10,
10,.5
34 34
3692 38
36 047.
47 .7
10,000,000,45
10 000 000.45
123,456,789,
10 102.3523
10,214 241

Expected Output (25 matches)

The first 25 lines should match your regex logic. All the other lines should not match.
Match each valid line using a named capture group called ‘match’.
(?<match>your regex logic)

Helpful Links

https://www.regular-expressions.info/possessive.html
https://www.regular-expressions.info/optional.html
https://www.regular-expressions.info/alternation.html

Asterisk the Gaul

HbDRS 🐴🐴

Preamble

Looks like the famous writer friend needs our help once again. This time around he wants you to reformat sections from his new comic book that he previously formatted using markdown style. Being the nice person you are, you aren't going to deny his request.

Description

To help the writer correct his formatting mistakes we are going to reformat words or strings of words that have one asterisk '*' on one side that corresponds to one asterisk '*' on the other side. Use the html tags <i>italic</i> to mark those text sections correctly as *italic text*.
Be aware that there are also sections with two asterisks '**', which should not be messed with.
Additionally there are single asterisks who have no corresponding partner-asterisk that also should not be matched by your logic.
Observe the below example carefully to understand the rules.

example:
Example with one * lonely asterisk, one *italic section* and one **section that is bold**
expected result:
Example with one * lonely asterisk, one <i>italic section</i> and one **section that is bold**

You need to use the built-in 'substitution' function of regex101.

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex + Count of characters of the substitution

Score-Self-Check

Fewest steps:

Good ~ 900 steps; Great ~ 500 steps; Amazing ~ 220 steps

Fewest chars:

Good ~ 90 chars; Great ~ 60 chars; Amazing ~ 45 chars

Hints

Lookahead and Lookbehind.

Challenge Data	Expected Output (9 matches)
This text is not italic.	This text is not italic.
This text is italic.	<i>This text is italic.</i>
This text is *partially* italic	This text is <i>partially</i> italic
This text has *two* *italic* bits	This text has <i>two</i> <i>italic</i> bits
bold text (not italic)	**bold text (not italic)**
**bold text with *italic* **	**bold text with <i>italic</i> **
part bold, *part italic*	**part bold,** <i>part italic</i>
*italic text **with bold** *	<i>italic text **with bold** </i>
italic **bold** *italic* **bold**	<i>italic</i> **bold** <i>italic</i> **bold**
*invalid markdown (do not parse)**	*invalid markdown (do not parse)**
random * asterisk	random * asterisk

Helpful Links

https://www.regular-expressions.info/lookaround.html

Markups, Markdowns, Markarounds

HbDRS 🐴🐴

Preamble

In the previous challenge we reformatted text that was in markdown-format into html-style. This time around we are going to validate markdown statements. Huge thanks go out to Damien Chillet (@d3.iso) who originally helped me out on making this challenge possible.

Description

I am going to cut the description short and actually move all the explanations to the 'Expected Output' section.

You need to use the built-in 'substitution' function of regex101.

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex + Count of characters of the substitution

Score-Self-Check

Fewest steps:
Good ~ 1500 steps; Great ~ 500 steps; Amazing ~ 300 steps
Fewest chars:
Good ~ 120 chars; Great ~ 85 chars; Amazing ~ 68 chars

Hints

Lookahead and Lookbehind. Character classes. Negated Character classes.

Challenge Data (match, no match)	Expected Output
[Basic link](http://example.com)	<a href="http://example.com">Basic link</a>
[Another](http://example.com/)	<a href="http://example.com/">Another</a>
Link: macr.ae	Link: <a href="https://macr.ae/">macr.ae</a>
[Text](https://test.this-test.com/)	<a href="https://test.this-test.com/">Text</a>
[Test!](https://this.com) hello	<a href="https://this.com">Test!</a> hello
I [I](https://TESTdomain.com) I	I <a href="https://TESTdomain.com">I</a> I
[number](http://0test.com/)	<a href="http://0test.com/">number</a>
[Invalid](http\\0test.com/)	[Invalid](http\\0test.com/)
[Invalid](invalid://example.com)	[Invalid](invalid://example.com)
[Invalid](mailto:nobody@example.com)	[Invalid](mailto:nobody@example.com)
[Invalid](javascript:alert())	[Invalid](javascript:alert())
[Invalid](http://test_ing.com)	[Invalid](http://test_ing.com)
[Invalid](http://inval.id,com)	[Invalid](http://inval.id,com)
![Image](http://example.com/cats.jpg)	![Image](http://example.com/cats.jpg)
![Other image](cats.jpg)	![Other image](cats.jpg)
I[radioactive](http://dolphin.com)	I[radioactive](http://dolphin.com)
[Invalid MarkDown](http://example.com)I	[Invalid MarkDown](http://example.com)I
[[cat-penguin](http://example.com)	[[cat-penguin](http://example.com)
[[Invalid MarkDown](http://example.com))	[[Invalid MarkDown](http://example.com))

Additional Explanations

Do not make your regex fail when encountering the word ‘invalid’.

Why should the bottom lines not be matched?

[Invalid](http\\0test.com/)
→ because it isn't well-known URL syntax (\\)

[Invalid](invalid://example.com)
→ because it isn't well-known URL syntax (invalid:)

[Invalid](mailto:nobody@example.com)
→ because it isn't well-known URL syntax (mailto)

[Invalid](javascript:alert())
→ because it isn't well-known URL syntax (javascript:alert())

[Invalid](http://test_ing.com)
→ because it isn't well-known URL syntax (underscore)

[Invalid](http://inval.id,com)
→ because it isn't well-known URL syntax (, instead of .)

![Image](http://example.com/cats.jpg)
→ because the '!' makes it an image-reference, therefore it shouldn't be converted into a hyperlink

![Other image](cats.jpg)
→ because the '!' makes it an image-reference, therefore it shouldn't be converted into a hyperlink

I[radioactive](http://dolphin.com)
→ because there is the letter 'I' without a space afterwards, which causes an invalid syntax

[Invalid MarkDown](http://example.com)I
→ because there is the letter 'I' without a space before it, which causes an invalid syntax

[[cat-penguin](http://example.com)
→ because there is a second '['

[[Invalid MarkDown](http://example.com))
→ because the parentheses mismatch '[...)', additionally this causes a syntax error

Helpful Links

https://www.regular-expressions.info/lookaround.html
https://www.regular-expressions.info/charclass.html
https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Roman Numerals

HbDRS

Preamble

This challenge marks the end of the 'medium' difficulty challenges. What better way to celebrate this than to make one that has absolutely no real-world application. Let's make a challenge about matching roman numerals, which is a superior numbering system used by Romans and Horses.

Description

In this challenge you need to come up with regex logic for matching roman numerals. Luckily, the logic behind roman numerals is known to mankind and well documented.

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex

Score-Self-Check

Fewest steps:
Good ~ 10000 steps; Great ~ 5500 steps; Amazing ~ 3700 steps
Fewest chars:
Good ~ 200 chars; Great ~ 100 chars; Amazing ~ 70 chars

Hints

Possessive Quantifiers. Optional Items. Alternation.

Challenge Data (match, no match)

I II III IV V VI VII VIII IX
X XIII XIV XV XVIII XIX
XXV XXVI XXVII XXVII XXIX
XXX XXXIII XXXVIII XXXIX
XL XLII XLV XLVIII XLIX
L LI LII LIII LIV LV LVI LVII LVIII LIX
LX LXVI LXIX LXXIX LXXXV LXXXVIII
XC XCV XCVI XCVII XCIX
C CI CV CIX
CXI CXX CXXX CXXXVIII CXXXIX CXLII CL CLV CLVIII CLX CLXX CLXXX CXC CXCIII CXCVII
CXCIX CC CCIII CCVIII CCIX CCXXV CCL CCLXXV CCCLXXV CD CDXXV CDL CDLXXV CDXC
D DIX DCLXVI DCLXXV DCCCXXVIII
CM CMLXXV
M ML MCV MCCCL MD MDCCXXV MDCCCLXXV MCML MCMXCVIII MCMXCIX
MM MMCCCXXV MMCDLXXV MMDL
MMM MMMCCXXVIII MMMCCCXXVIII MMMD
MMMCMXCV MMMCMXCVIII MMMCMXCIX
VV VVX XVV IIII IIIII IVIVIV XIIX XXXXII VC IIXV IC CXIIL
LVIXXX LIXXX XXXVIIII XXXVV MCVV MLTK IIV IIX IIL IIM IIC CCM
CMD CMDB L0L LOL GG RAF WTF XIIIXCVIIVMC MMIXIII CCCXXXIIX
MCMM MDCCLXXVIIII CCCD CCCXXXIIII MMMMXXXX

Expected Output (111 matches)

Using the logic you came up with you need to match every single numeral marked in green. While simultaneously not matching the red ones.

Helpful Links

https://www.regular-expressions.info/possessive.html
https://www.regular-expressions.info/optional.html
https://www.regular-expressions.info/alternation.html
https://www.factmonster.com/math-science/mathematics/roman-numerals

Vali-Dates

HbDRS

Preamble

Now we finally arrive at the hard challenges. Congratulations to you if you have come this far. Remember matching the correct dates for the 29th of February a couple of challenges back? This time I want you to validate dates that follow a certain scheme.

Description

The challenge data includes a list of dates with different formats.

I want you to match dates with the following two formats:
yyyy/mm/dd HH:MM
yyyy/mm/dd HH:MM:SS

All the other formats and especially invalid dates should not be matched.

Do not validate if a month has 28 or 29 days (leap years) or 30 or 31 days.
Do however validate if there are obvious errors like 27:44:13 or 2021/17/25.

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex

Score-Self-Check

Fewest steps:
Good ~ 1200 steps; Great ~ 700 steps; Amazing ~ 400 steps
Fewest chars:
Good ~ 140 chars; Great ~ 100 chars; Amazing ~ 85 chars

Hints

Alternation. Character classes.

Challenge Data (match, no match)

```
2012/09/18 12:10
2001/09/30 23:59:11
1995/12/01 12:12:12
1001/01/07 14:27
2021/10/20 10:10
2000/01/01 01:01:01
2007/07/22 22:34:59
2021/05/05 00:00:00
2021/9/18 23:40
2013/XY/09 09:09
2021/00/01 01:49:59
2012/13/25 22:17:00
1994/11/00 12:12
2012/12/4 12:12
2009/11/11 24:00:00
2021/06/24 13:60
2002/10/10 14:59:60
a2021/11/11 11:11:11
2005/05/05 05:05:05d
2000 01 01 01:01:01
2007-07-22 22:34:59
2020/05/05 00/00/00
```

Expected Output (8 matches)

The first 8 lines should match your logic, while the rest of the lines should not be matched.

Helpful Links

https://www.regular-expressions.info/alternation.html
https://www.regular-expressions.info/charclass.html

Credit Card Numbers

HbDRS 🐴🐴🐴

Preamble

As quickly as we arrived at the hard challenges we are going to leave them behind soon, so we can head towards the insane ones. This challenge serves as some sort of gate-keeper. If you are able to master this one you are ready for the insanity that comes next.

Description

I want you to match credit card numbers (CCN) following these **six** simple rules.

Rule #1: The CCN **must** start with a 4, 5 or 6.
Rule #2: The CCN **must** only contain numbers 0 to 9 and optionally hyphens.
Rule #3: The CCN **must** contain exactly 16 digits... no less, no more.
Rule #4: The CCN **may** come in groups of 4 (four) digits separated by a hyphen. There must be either three hyphens in total or none at all. Nothing in between.
Rule #5: The CCN **must not** use any other separator that is different from a hyphen.
Rule #6: The CCN **must not** have 4 (four) or more consecutive repeated digits.

While rules #1 to #5 shouldn't pose much of a challenge to you, rule #6 is more difficult to be implemented correctly.

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex

Score-Self-Check

Fewest steps:
Good ~ 5000 steps; Great ~ 3500 steps; Amazing ~ 1100 steps
Fewest chars:
Good ~ 130 chars; Great ~ 90 chars; Amazing ~ 60 chars

Hints

Alternation. Character classes. Back References. Optional Items.

Challenge Data (match, no match)

4123456789123456
5123-4567-8912-3456
6000700080009000
6653625879615786
4424424424442444
6543-6543-6543-6543
2223334445556660
61234-567-8912-3456
5133-3367-8912-3456
5123 - 3567 - 8912 - 3456
4512-1234 - 1244 -3256
5553-323519230091
55533235-19230091
555332351923-0091
abcd-eFgh-Hjkl-mnop
42536258796157867
4424444424442444
5122-2368-7954 - 3214
44244x4424442444
0525362587961578
4332-2223-5532-2010
5522,5522,5522,5522
5522_5522-5522,5522
6543-6543 6543-6543
5533 5555 5522 2255
5553-5553-5553-5555
8231-9200-2724-2219
6333-3444-2221-1133
5554-4433-3222-111O

Expected Output (6 matches)

The first 6 lines should match your logic, while the rest of the lines should not match.

Valid CCN Examples:

4253625879615786

4424424424442444

5122-2368-7954-3214

Invalid CCN Examples:

42536258796157867	17 digits in card number
4427777724442444	Consecutive digits are repeating 4 or more times
5122-2368-7954 - 3214	Spaces between the hyphen separator
44244x4424442444	Contains non-digit character
0525362587961578	Doesn't start with 4, 5 or 6

Helpful Links

https://www.regular-expressions.info/alternation.html
https://www.regular-expressions.info/charclass.html
https://www.regular-expressions.info/backref.html
https://www.regular-expressions.info/backref2.html
https://www.regular-expressions.info/optional.html

The Picky Painter

HbDRS 🐴🐴🐴🐴🐴

Preamble

With all the weird stuff going on in the world of modern art, there is an artist who has a big art collection consisting of paintings only in grayscale. People love it. You are her loyal apprentice who has no clue about modern art, but helps her to find the right colors for her next masterpiece. She likes grayish colors. Dark gray, light gray, gray gray, all shades of gray. Fifty.

However she already has decided to use a subset of them for her next painting and wants you to select them from a long list of colors. Unfortunately, this list is rather ugly and looks like a copy'n paste job gone wrong. There are some other colors and issues with incorrect formatting.

Description

The objective is fairly simple. Just match the gray colors.

How are gray colors defined?
Gray colors just have the same percentage of red, green and blue or alternatively cyan, magenta and yellow coloring to it. But not 100% (white) or 0% (black). Additionally look for mistakes in color notation and correct formatting. Your matching colors need to be captured in a named capturing group called "match".

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex

Score-Self-Check

Fewest steps:
Good ~ 3500 steps; Great ~ 2000 steps; Amazing ~ 1100 steps
Fewest chars:
Good ~ 370 chars; Great ~ 280 chars; Amazing ~ 210 chars

Hints

Oftentimes it is useful to start from scratch. Save what you have and start anew.

Challenge Data (match, no match)

#111
#aaa
#eEe
#111111
#6F6F6F
#efEfEF
rgb(2, 2, 2)
rgb(15,15,15)
rgb(2.5, 2.5,2.5)
rgb(1, 01, 000001)
rgb(20%, 20%,20%)
rgba(4,4,4,0.8)
rgba(4,4, 4,1)
rgba(3,3,3,0.12536)
rgba(10%,10%,10%,5%)
hsl(20,0%, 50%)
hsl(0, 10%, 100%)
hsl(0.5, 10.5%, 0%)
hsl(5, 5%, 0%)
hsla(20, 0%, 50%, 0.88)
hsla(0, 0%, 0%, 0.25)
#ef4
#eEf
#11111e
#123456
rgb(2, 4, 7)
rgb(10, 10,100)
rgb(1.5%, 1.5%, 1.8%)
rgba(1, 01, 0010, 0.5)
hsl(20, 20%, 20%)
hls(0 1% 01%)
hsla(0, 10%, 50%, 0.5)
#11111
#000000000
rbb(1, 1, 1)
rgb(10, 10, 10, 10)
rgb(257, 257, 257)
rgb(10%, 10, 10)
hsl (20,0%, 500)
argb(1.1.1)

Expected Output (21 matches)

The first 21 lines should match your logic, while the rest of the lines should not match.

Match each valid line using a named capture group called ‘match’.
(?<match>your regex logic)

Remember that your regex only has to work for the presented data. The challenge is not about validating all the possible coloring formats or all the grayish-colors out there.

Helpful Links

https://en.wikipedia.org/wiki/Color_theory

Revelation

HbDRS 🐴🐴🐴🐴🐴

Preamble

This challenge marks the last one of this collection. The author has never shown this challenge to anyone before. This challenge demands you to use everything you have learned so far. It is diabolically difficult. At least that is what the author hopes for.
The original idea for this challenge came from https://www.reddit.com/user/jordanreiter.

Description

Ever had issues with reformatting weird excel spreadsheets to get them into a working csv format? No? Yes? Maybe?
Whatever your answer to this question might be, you just have to achieve **one simple goal**.

Get the presented *challenge data into the form of the expected output.*
Get the presented ***challenge data into*** *the form of the expected output.*
Get the presented challenge data into ***the form of*** *the expected output.*
Get the presented challenge data into the form of ***the expected output***.

You need to use the built-in 'substitution' function of regex101.

Check the 'Expected Output' section for further clarification.

Rules

Default rules.

Winning Categories

Fewest steps.
Count of steps displayed on regex101.com
Fewest chars.
Count of characters of the regex + Count of characters of the substitution

Score-Self-Check

The author has achieved a solution with 2104 steps and 94 chars. Can you beat him?

Hints

Think you have solved it? Better check it vigorously before you call your solution done. There are lots of small traps laid out in it to throw you off and ruin your day.

Challenge Data

```
This is a test
This is another test
This "big test" is a test
This "big test" is a "big test",yeah!
Almost "this entire" thing "is just a" quote
Matthew Banana-Horsey's friend Joseph
Matthew Banana-Horsey is a test
Matthew Banana-Horsey is--a--test
This----is--test
This-----is-test field
Don't say anything,ok?
I can't think
This " is a " test
don't tell Matthew Banana-Horsey that I broke Brandon's toy horse
I can't see Matthew Banana-Horsey anywhere; can you?
Too long; didn't read
Matthew Banana-Horsey's car was stolen
Damien Chillet is a regex prodigy
```

Expected Output

```
This,is,a,test
This,is,another,test
This,big test,is,a,test
This,big test,is,a,big test,yeah,!
Almost,this entire,thing,is just a,quote
Matthew,Banana-Horsey's,friend,Joseph
Matthew,Banana-Horsey,is,a,test
Matthew,Banana-Horsey,is,a,test
This,is,test
This,is-test,field
Don't,say,anything,ok,?
I,can't,think
This, is a ,test
don't,tell,Matthew,Banana-Horsey,that,I,broke,Brandon's,toy,horse
I,can't,see,Matthew,Banana-Horsey,anywhere,can,you,?
Too,long,didn't,read
Matthew,Banana-Horsey's,car,was,stolen
Damien,Chillet,is,a,regex,prodigy
```

Additional Explanations

Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. ***Get the presented challenge data into the form of the expected output.*** *Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output.* ***Get the presented challenge data into the form of the expected output.*** *Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output.* ***Get the presented challenge data into the form of the expected output.*** *Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output.* ***Get the presented challenge data into the form of the expected output.*** *Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output. Get the presented challenge data into the form of the expected output.*

Helpful Links

None.

Acknowledgements

Cary Petterborg

Cary Petterborg inspired me to learn regex when I joined the slack community in 2017. He not only showed me how to improve my skills, but also encouraged me to try out new approaches and look at problems from different angles. Without him I would've never mastered regex.

Find Cary on LinkedIn: https://www.linkedin.com/in/carypetterborg/

Dal Jeanis

Dal Jeanis always believed in me and picked me up from the ground every time I had tripped up. He saw the potential in me and was able to convince me to keep fighting against all odds. His vast knowledge and professional attitude were beneficial in my endeavour of becoming who I am today.

Find Dal on LinkedIn: https://www.linkedin.com/in/daljeanis/

Callum Macrae

Callum Macrae's website callumacrae.github.io/regex-tuesday/ gave me the idea to host periodically occurring regex challenges in the Splunk> community. The challenges on his website are the baseline for most challenges in this collection.

Check out Callum on: https://macr.ae/

The amazing regex web resources

Thanks go out to the amazing regex resources on the web. I linked to many of them in this guide. Special thanks go out to **Firas Dib**, the creator of regex101.com, a site which made hosting regex challenges manageable. Besides that, regex101.com is my go-to site whenever I feel like doing regex.

The amazing folks from the unlimited_randomness channel

Thanks to all the amazing people from the ‘unlimited_randomness’ channel on the splunk user group slack. Thank you for amazing conversations, funny stories and great entertainment altogether. Without you all I wouldn’t have made it so far and probably gone insane by now.

All the challenge participants

I want to thank each and everyone of you who participated in my regex challenges in the past. Without your contribution it wouldn’t be possible to now publish this collection. Big thanks to everyone for spending their valuable time. I hope you all had fun doing so.

The people proof-reading this document

Many thanks go out to the awesome folks that helped me finalize the contents of this document. They spent their free time hunting through this document for spelling errors, issues with grammar or factual mistakes.

Family and friends

Last but not least I want to thank my family and friends who supported me on countless endeavours throughout my life and hopefully will continue doing so in the future.

About the author

Matthias Kowalewski, the author of this document, is, despite popular belief, not a real horse. He loves regex as much as being random.

If he is not actively hosting regex challenges in the Splunk> community he works as a Splunk> Consultant with focus on IT-Security. He also enjoys playing strategy games on his computer or hacking into vulnerable machines that were set up in his private security lab.

Find Matthias on LinkedIn: www.linkedin.com/in/matthiaskowalewski

About creating this document

It was a lot of fun, and hard work, for me to put this collection of regex challenges together. This is my first publication in the form of an e-book and I did learn a lot throughout the creation process.
I am proud of how it all turned out in the end. I can only hope that you'll like it as much as I do and that the newly gained knowledge about regular expressions will help you in your career. Although, I already know that it will certainly be beneficial.

Contact me for feedback!

Wanna share your score?
Has this guide helped you to get better at regex?
Curious about how other people did in the challenges?
Wanna see my cool solution I use to validate IPv4 addresses?
Do you want to tell me about how you tortured your coworkers with the challenges?
Anything else regex related?

If yes, then please contact me at: horsefez@pm.me

Want me to solve your regex problems at work?
I am not going to. Sorry.
I am not paid to do your job.
GO AND LEARN REGEX. LOL.