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

   The regex /abababa/ looks for an identical match in terms of characters inserted i.e. it matches the exact term given.

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

   The regex /ab+c?/ matches any term which starts with "a", followed by one or more "b"s with an optional character "c".

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

   The regex /a.[bc]+/ matches any term starting with the character "a" followed by any character, and then it matches one or more items from a list containing characters "b" and "c".

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

   The regex /abc|xyz/ matches the terms containing the following characters by order “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.?

The regex /[a-z]+[\.\?!]/ matches one or more non capitalised english alphabet characters followed by either a dot, question mark, or an exclamation mark.

6. ***Which of the following matches regex /[a-zA-Z]*[^,]=/***

1. Butt=
2. BotHEr,=
3. Ample
4. FIdDIE7h=
5. Brittle =
6. Other.=

The regex /[a-zA-Z]*[^,]/ matches zero or more alphabet characters(both capitalised and not), followed by any character that isn't a comma(","), and ending with an equal symbol("=").

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

The regex /[a-z][\.\?!]\s+[A-Z]/ matches any non capitalised english alphabet character followed by either a dot, question mark or exclamation mark, then followed by one or more spaces and ends with a capitalised english alphabet character.

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

The regex /(very)+(fat)?(tall|ugly) man/ matches the following:
The word very and a space one or more times, followed by optional term fat and a space followed with the word "tall" or "ugly" and ends with a space followed by the word "man".

9. ***Which of the following matches regex /<[^>]+>/?***

1. <an xml tag>
2. <opentag> <closetag>
3. </closetag>
4. <>
5. <with attribute="77">

The regex /<[^>]+>/ matches any pattern which starts with a less-than sign, followed by one more more characters that are not a greater-than sign, and ends with a greater-than sign.

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])|(?:[1-2]\d)|(?:3[0-1]))\/((?:0[1-9])|(?:1[0-2]))\/((?:0{
3}[1-9])|(?:0{2}[1-9]\d)|(?:0{1}[1-9]\d{2})|(?:[1-9]\d{3}))$/gm
```

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])|(?:[1-2]\d)|(?:3[0-1]))(\/|.)((?:0[1-9])|(?:1[0-2]))\2((?:
3}[1-9])|(?:0{2}[1-9]\d)|(?:0{1}[1-9]\d{2})|(?:[1-9]\d{3}))$/gm
```