1. Background Training Phase:

1.1. Introduction to Regex Syntax:

REGEX	MATCHES
`ab`	'ab'
`a|b`	'a' or 'b'
`a*`	'', 'a', 'aaa', ... - zero or more 'a'
`a+`	'a', 'aaa', ... - one or more 'a'
`a?`	'', 'a' - zero or one 'a'
`[a-z]`	'a', 'x', 'z', ... - any character in the range a-z
`\w`	[0-9a-zA-Z_] - any word character, including alphanumeric characters and '_'

Examples:

`a(bc)*`	matches a string that has *'a'* followed by zero or more copies of the sequence '*bc'*
`a(b|c)`	matches a string that has '*a'* followed by *'b'* or followed by *'c'*

1.2. Infinite Ambiguity:

Ambiguous regex:
A regex is ambiguous when, for the same input, there are multiple ways to match it.
For example: for regex `(a|a)`, there are two ways to match the input 'a' (with the first 'a', or with the second 'a' of the regex).

Infinitely ambiguous regex**:**
A regex is infinitely ambiguous when the number of ways to match an input grows polynomially or exponentially with the length of the input.
For example: for regex `(a|a)*` and input 'aaaa….', there are two ways to match each *'a'* of the input, making the total number of ways to match the whole input grow exponentially with its size.

In this study, we will evaluate different ways to identify infinite ambiguity in regexes.

1.3. Some Terminology for This Study:

Node: An atom of the regex. For example, in the regex `(a|ab)`, 'ab' is a node.
Quantifier: **'***'* ,'**+***', '?', '{n}', '{n,m}', {n}* are quantifiers.

Quantified node: If a node has a quantifier, it is called a quantified node. For example, in the regex `a*b*`, a* is a quantified node.

Overlap: Overlap means that there is an intersection between the strings matched by the nodes.

`a` <-> `a` - there is an intersection: ‘a’

`a` <-> `ab` - there is no intersection

`[a-c]` <-> `[b-d]` - there are multiple intersections: ‘b’, and ‘c’

2. Anti-patterns & Tasks:

2.1. Anti-patterns 1:

Please read the following anti-patterns, which will help you identify and fix an **infinitely ambiguous** regex:

QOA (Quantified Overlapping Adjacency): An example of this anti-pattern is: `/\w*#?\w*`. The two quantified \w* nodes overlap, and are adjacent because one can be reached from the other by skipping the optional octothorpe.

From each node we walk forward looking for a reachable quantified adjacent node with an overlapping set of characters, stopping at the earliest of: a quantified overlapping node (QOA), a non-overlapping non-optional node (no QOA), or the end of the nodes (no QOA).

QOD (Quantified Overlapping Disjunction): An example of this anti-pattern is `/(\w|\d)+/`. Here we have a quantified disjunction (/(...|...)+/), whose two nodes overlap in the digits, 0-9.

Star height > 1: An example of this anti-pattern is `/(a+)+/`
To measure star height, we traverse the regex and maintain a counter for each layer of nested quantifier: +, *, and check if the counter reached a value higher than 1. In such cases, the same string can be consumed by an inner quantifier or the outer one, as is the case for the string “a” in the regex `/(a+)+/`.

2.2. Task set-1:

Please complete the following tasks, answering with a regex that is NOT infinitely ambiguous. Use the anti-patterns above to guide you.

You have 5 minutes for each task

1. Write one regex that matches (consider full match only) one or more ‘b’ followed by a single ‘c’:
   Example matching strings: bc, bbc, bbbbc, bbbbbc, bbbbbbbbbbbbbbbbbbbbbc
   Example non-matching strings: namebcname,

2. Write one regex that matches (consider full match only) **one or more repetitions** of the following: *one or more ‘b’ followed by a single ‘c’*
   Example matching strings: bcbc, bbcbbcbbc, bbbbbcbbbbbc, bbbbbbbbbbbbbbbbbbbbcbbbbbbbbbbbbbbbbbbbbc

3. Write a regex to match (consider full match only) one or more ‘a’ or ‘b’, followed by one or more repetitions of ‘ab’

   Example matching strings: aab, bab, aaab, aaaaab, bab, bbbab, aaaababababab, bbbbababababab

4. Write a regex to match one or more occurrences of the strings ‘a’, ‘b’, or ‘ab’.
   Example matching strings: aaaaaaaaaa, bbbbbbbbbbb, abababababababababab

5. Write one regex that matches (consider full match only) one or more ‘a’ followed by an optional b followed by one or more ‘a’
   Example: aaaabaa, aaaaa, abaaaa

2.3. Antipatterns 2:

Please read the following anti-patterns which will help you identify and fix an infinitely ambiguous regex.

Concat 1: R = ...P*Q*... (R has a sub-regex P*Q*): The two quantified parts P* and Q* can generate some shared string s.
Example: `a*(aa)*`, both can generate 'aa'
`(x|y)*(xy)*` : both can generate 'xy'.

Concat 2: R = ...P*SQ*... (R has a sub-regex P*SQ*): The two quantified parts P* and Q* can generate a string s from the middle part S.
Example: `a*aa*` : The quantified parts are a* which can generate the middle part 'a'
`(a|b)*ab(ab)*` : The quantified parts (a|b)* and (ab)* can generate the middle part 'ab'.

Concat 3: R = ...P*S*Q*... (R has a sub-regex P*S*Q*): Advanced form of Concat 1. Since S* includes an empty string, the ambiguity between P* and Q* can be realized.
Example: `a*b*a*`, b* can be skipped

Star 1: R*, R= (P|Q|...): There is an intersection between any of the two alternates i.e., both can generate some shared strings.

Example: `(\w|\d)*`, both can generate digits [0-9]

Star 2: R*, R= (P|Q|...): You can make one option of the alternation by repeating another option multiple times or by concatenating two or more options multiple times.

Example: `(x|xx)*` , we can create the 2nd option 'xx' by repeating the first option 'x' twice.
`(x|y|xy)*` , we can create the 3rd option 'xy' by adding the first option 'x' and second option 'y'.

Star 3: R*, R= (...P*...): Nested quantifiers, but only if RR would follow any of the concat antipatterns above.

Examples:
For `(a*)*`, we write a*a*, it is infinitely ambiguous by **Concat 1.**
For `(xy*)*`, we write xy*xy*, it is not infinitely ambiguous by any of the concat anti-patterns.

2.4. Task set-2:

Please complete the following tasks, answering with a regex that is NOT infinitely ambiguous. Use the anti-patterns above to guide you.

You have at most 5 minutes for each of the task

6. Write one regex that matches (consider full match only) one or more ‘b’ followed by a single ‘c’:
   Example matching strings: bc, bbc, bbbbc, bbbbbc, bbbbbbbbbbbbbbbbbbbbbc
   Example non-matching strings: namebcname,

7. Write one regex that matches (consider full match only) **one or more repetitions** of the following: *one or more ‘b’ followed by a single ‘c’*
   Example matching strings: bcbc, bbcbbcbbc, bbbbbcbbbbbc, bbbbbbbbbbbbbbbbbbbbcbbbbbbbbbbbbbbbbbbbbc

8. Write a regex to match (consider full match only) one or more ‘a’ or ‘b’, followed by one or more repetitions of ‘ab’

   Example matching strings: aab, bab, aaab, aaaaab, bab, bbbab, aaaababababab, bbbbababababab

9. Write a regex to match one or more occurrences of the strings ‘a’, ‘b’, or ‘ab’.
   Example matching strings: aaaaaaaaaa, bbbbbbbbbbb, abababababababababab

10. Write one regex that matches (consider full match only) one or more ‘a’ followed by an optional b followed by one or more ‘a’
    Example: aaaabaa, aaaaa, abaaaa