Lex regular expressions

abc	: The abc string.
“void”	: The void string.
\”	: A double quote character.
\/	: A slash character.
[abc]	: a, b or c. Same as a|b|c.
[a-zA-Z]	: Any alphabetical character.
[1-9]	: A non zero digit.
[+-]?	: +, - or neither. Same as +|-|ε.
(ab)?b	: abb or b. Same as (abb)|b.
[0-9]+	: A non empty sequence of digits.
[0-9]*	: Any number of digits in a row. Same as ([0-9]+)?.
[^a]	: Any character except a.
[^abc]	: Any character except a, b or c.

[ ]	: A space character.
[\t\n]	: Any form of whitespace. (A space, a tab or a line feed.)
.	: Any character except \n. Same as [^\n].
\.	: A dot character.
‘\\n’	: The exact following string : ‘\n’

0x(0|([1-9a-fA-F][0-9a-fA-F]*))

A hexadecimal number starting with 0x. Example : 0xFFFF or 0xaB42c3 or 0x0.

Note that when writing regular expressions you should avoid using whitespaces with the intent of making the code more readable.

The below expression is not equal to the one stated above.

0x(0 | ([1-9a-fA-F] [0-9a-fA-F]*))

(r1)/(r2)   : r1 only if followed by r2.

Note : This looks ahead to match the rule but does not actually consume the lookahead characters. So only r1 is returned.

Example :

1. {integer}/[=]
2. “==”

Input : 100==20

100 is returned as a token by rule 1.

== is returned as a token by rule 2.

20 is not returned as a token as it is not followed by an = character.