Concepts of Programming Languages, CSCI 305, Fall 2021
Regular Expressions, MiniC, Sept. 10

Updated: Oct. 11, 2021

MiniC Programs

According to the language description "' is a legal character, but would confuse the lexical analyzer. Also, the string "xy"xy" is allowed, but the lexical analyzer would not see it as the user probably intended. Similarly, /*xy*/xy*/ is allowed, but the lexical analyzer would not see it as the user probably intended. These regular expressions restrict these sequences. Don't worry if your regular expressions allow these sequences. These sequences will not be included in the test data. (In these regular expressions, once the second * is seen in a comment, the next character must be a /, closing the comment.)

Characters that will need a column in the final table:
a-z | A-Z | 0-9 | . | , | _ | + | - | * | \ | < | > | ! | = | / | ' | " | (|) | { | } | ;
0x20 (space) |
0x09 (tab) |
0x0a (carriage return) |
0x0d (new line)|

Helpers:
alpha → a-z | A-Z
octal → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
digit → octal | 8 | 9
nonzero → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
hex → digit | a | A | b | B | c | C | d | D | e | E | f | F
period → .
comma → ,
semicolon → ;
space → the character for 0x20
tab → the character for 0x09
cr → the character for 0x0a
nl → the character for 0x0d

most → alpha | digit | period | comma | _ | + | - | \ | < | > | ! | = |
/ | (|) | { | } | semicolon

whiteSpace → space | tab | cr | nl

escapeSequence → \ ∘ (\ | ' | " | a | b | f | n | r | t | v)

comment → / ∘ * ∘ (most | ' | " | whitespace)* ∘ * / Hopefully the reader can distinguish between * and the Kleene closure meta-symbol

decimalOnly → digit* (. digit | digit .) digit*
exponentOnly → (digit ∘ digit*) ∘ (e | E) ∘ (+ | - | ε) ∘ (digit ∘ digit*)

both → digit* (. digit | digit .) digit* ∘ (e | E) ∘ (+ | - | ε) ∘ (digit ∘ digit*)

Regular Expressions:

Token Name	Regular Expression
addOp	+ | -
assignOp	(ε | + | - | * | /) ∘ =
boolLiteral	Recognized as an identifier; lookup t, f, T or F in lookup table and return boolLiteral
char	' ∘ (most | " | * | escapeSequence) ∘ '
comma	,
comparator	[(< | >) ∘ (= | ε)] | [(= | !) ∘ =]
floatLiteral	decimalOnly | exponentOnly | both
incrOp	(+ ∘ +) | (- ∘ -)
identifier	(alpha | _) ∘ (alpha | _ | digit) *
intLiteral	([0 ∘ (x | X)) ∘ hex ∘ hex*] | [0 ∘ octal ∘ octal*] | [nonZero ∘ digit*]) ∘ [((u | U | ε) ∘ (l | L | ε)) | ((l | L | ε) ∘ (u | U | ε))]
lbrace	{
logicalNot	!
lparen	(
multOp	* | /
rbrace	}
rparen	)
semicolon	;
string	" ∘ (most | ' | * | 0x20)* ∘ "

Lookup words:
t, f, T, F, break, bool, case, char, const, continue, default, do, double, else, false, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, stuct. switch, typedef, union, unsigned, void, volatile, while, _Packed