Little Ole Language (LOL) Syntax Rules in EBNF Notation

A. *Tokens are indicated in bold-face between double quotes; semi-tokens are indicated in non-bold capitals; nonterminals are indicated between angular brackets.*

```
---------- Standard beginning for the set of grammar rules -------------


0.  <S> ::= <program>


---------- Syntax rules for program -------------------------------


1.  <program> ::= { <declaration> }
                  { <statement> }


---------- Syntax rules for declarations --------------------------


2.  <declaration> ::=  <type> IDENT

3.  <type> ::= <simple_type> | <array_type>

4.  <simple_type> ::=   “int” | “float” | “char”

5.  <array_type> ::=  “array” <simple_type> “[“ INTLIT { "," INTLIT } “]”


------------ Syntax rules for statements ------------------------


6.  <statement> ::=
      ( <input_stmt> |
        <output_stmt> |
        <assignment_stmt> |
        <if_stmt> |
        <while_stmt> )

7. <input_stmt> ::= “read” “(” <designator> “)”

8. <output_stmt> ::= “write” “(” [ <expression> { “,” <expression> } ] “)”

9. <assignment_stmt> ::= <designator>  “=” <expression>

10. <if_stmt> ::=
      “if”  “(“ <expression> “)”
        “{” { <statement> }
      “}”

11.  <while_stmt> ::=
      “while” “(“ <expression> “)”
        “{” { <statement> }
      "}"
```

```
------------ Syntax rule for designator ------------------------------

12.  <designator> ::= IDENT [ “[” <expression> { “,” <expression> } “]” ]

------------ Syntax rules for expressions ---------------------------

13.  <expression> ::= <simple_expr> [ <relational_op> <simple_expr> ]

14.  <relational_op> ::= “==” | “!=” | “<” | “<=” | “>” | “>=”
Note:!= is not equal)

15.  <simple_expr> ::= [ <unary_op> ] <term> { <add_op> <term> }

16.  <unary_op> ::= “_”
(Note: _ is the underscore character)

17.  <add_op> ::= “+” | “-” | “|”

18.  <term> ::= <factor> { <mult_op> <factor> }

19.  <mult_op> ::= “*” | “/” | “//” | “%” | “&”
(Note: These are: multiply, divide, integer divide, modulo, logical AND)

20.  <factor> ::= INTLIT | FLOLIT | CHRLIT | STRLIT |
                  <designator> | “(” <expression> “)” | “~” <factor>
(Note:  ~ is logical NOT)
```

NOTES:

1. The definitions for IDENT, CHRLIT, STRLIT, INTLIT, and FLOLIT are below.

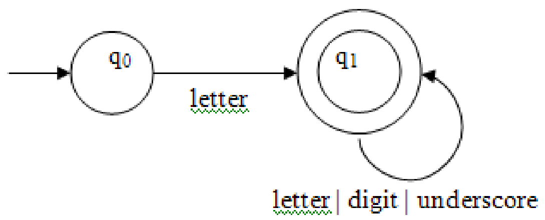
2. Note the following Operator Precedence Table:

Operator example	**Order**	**Meaning**
bigNum [10]	applicative	array index
~	prefix	logical NOT
* / // % &	infix	multiplicative
+ - |	infix	additive
_	prefix	unary minus
== != < <= > >=	infix	logical relations

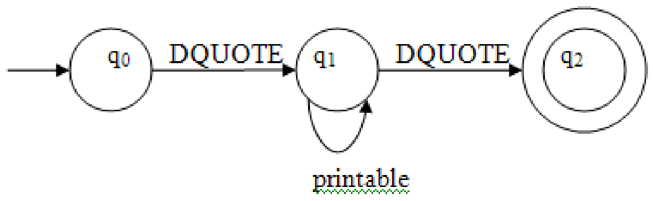
Operators are listed in order of decreasing binding power. All infix operators are left associative; exponentiation (if this semester’s language has an exponentiation operator) and sign are right associative. Note that **parentheses** take precedence over everything else and can be used to override the precedence rules.

Finite state automata for “semi-tokens”:

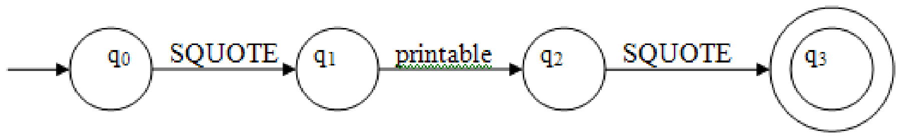
IDENT



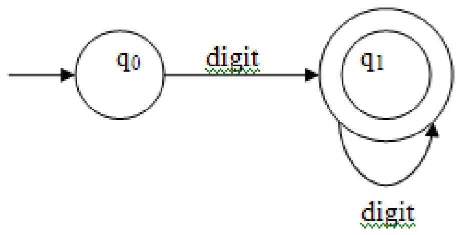
STRLIT



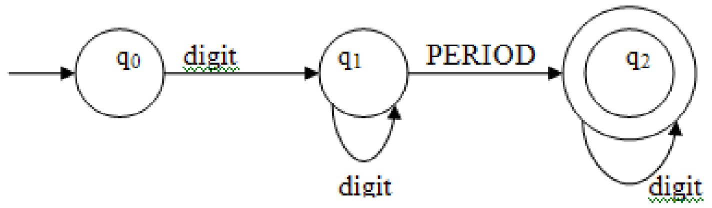
CHRLIT



INTLIT



FLOLIT


q0
digit
q1
PERIOD
q2
digit
digit