**Alphabet:**

**a. Upper (A-Z) and lower case letters (a-z) of the English alphabet**

**b. Underline character '\_';**

**c. Decimal digits (0-9);**

**- operators + - \* / % = < <= == >= || && ! !=**

**- separators { } ; space () .**

**- reserved words:**

**list num logic for from to if else maximum get print true false at by**

**b. identifiers**

**-a sequence of letters and  digits, such that the first character is a letter; the rule is:**

**<identifier> ::= <letter> | <identifier><letter> | <identifier> <digit>**

**<letter> ::= A | B | ... | Z | … | a | … | z**

**<digit> ::= 0 | 1 |...| 9**

**c. constants**

**1.integer :**

**<numconst> ::= +<num> |-<num> |<num> | 0**

**<num> ::= <non-zero-digit> | <num> <digit>**

**<non-zero-digit> ::= 1 | 2 | … | 9**

**2.character**

**<symbol> = . | space | - | ? | ! | , | ‘ | “ | ;**

**<character> ::= <letter>|<digit>|<symbol>**

**3.string**

**<string> ::= <character> | <string> <character> | empty**

**4. logic const**

**<logic> ::= true | false**

**2. Syntax**

**<program> ::= <decllist> ; <cmpdstmt>**

**<decllist> ::= <declaration> | <declaration> ; <decllist>**

**<declaration> ::= <type> IDENTIFIER**

**<type> ::= logic | list | num**

**<cmpdstmt> ::= <stmt> | <stmt> ; <cmpdstmt>**

**<stmt> ::= <simplstmt> | <structstmt>**

**<simplstmt> ::= <assignstmt> | <iostmt>**

**<assignstmt> ::= IDENTIFIER := <expression>**

**<expression> ::= <expression> + <term> | <term> | <expression> - <term>**

**<term> ::= <term> \* <factor> | <term> / <factor> | <term> % <factor> | <factor>**

**<factor> ::= ( <expression> ) | IDENTIFIER | CONSTANT**

**<iostmt> ::= get | print ( IDENTIFIER ) | print ( CONSTANT )**

**<structstmt> ::= <cmpdstmt> | <ifstmt> | <forstmt>**

**<ifstmt> ::= <ifbody> <stmt> | <ifbody> <stmt> else <stmt> | <ifbody> {<cmpdstmt>} | <ifbody> {<cmpdstmt>} else {<cmpdstmt>}**

**<ifbody> ::= if ( <condition> )**

**<forstmt> ::= <forbody> <stmt> | <forbody> {<cmpdstmt>}**

**<forbody> ::= for <declaration> from integer to integer | for <declaration> from integer to integer by integer | for IDENTIFIER from integer to integer | for IDENTIFIER from integer to integer by integer**

**<condition> ::= <expression> RELATION <expression>**

**<RELATION> ::= < | <= | == | != | >= | >**

**TOKENS**

**+**

**-**

**\***

**/**

**%**

**<**

**<=**

**=**

**==**

**!=**

**>=**

**>**

**&&**

**||**

**!**

**{**

**}**

**(**

**)**

**;**

**.**

**num**

**list**

**logic**

**for**

**from**

**to**

**if**

**else**

**maximum**

**get**

**print**

**true**

**false**

**at**