

```

1  Alphabet:
2      a. Upper (A-Z) and lower case letters (a-z) of the English alphabet
3
4      b. Underline character '_'
5
6      c. Decimal digits (0-9)
7
8  Lexic:
9
10     a.Special symbols, representing:
11
12         - operators
13           + - * / = < <= == >= and or
14
15         - separators
16           () [] {} , ;
17
18         - reserved words:
19           let bool int string read write if then for while do true false
20
21     b.identifiers
22
23         -a sequence of letters and digits, such that the first character is a letter; the rule
24         is:
25
26             identifier = (letter | "_") {character}
27             character = letter | digit | "_"
28             letter = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
29             digit = "0" | nonzero
30             nonzero = "1" | "2" | ... | "9"
31
32     c.constants
33
34         1.integer:
35
36             integer = "0" | ["+" | "-"] nonzero {digit}
37
38         2.boolean:
39
40             boolean = "true" | "false"
41
42         3.string:
43
44             string = "" {character} ""

```

```
1  #
2  +
3  -
4  *
5  /
6  %
7  =
8  <
9  <=
10 ==
11 >=
12 and
13 or
14 (
15 )
16 [
17 ]
18 {
19 }
20 ,
21 ;
22 let
23 bool
24 int
25 string
26 read
27 write
28 if
29 then
30 for
31 while
32 do
33 true
34 false
```

```

1  program = "#" block "#"
2  block = statement ";"
3
4  statement = compoundStatement | declarationStatement | assignmentStatement | ioStatement |
ifStatement | loopStatement
5  compoundStatement = statement ";" statement
6
7  declarationStatement = "let" declaration
8  declaration = type declaree {"," declaree} | declaration "," declaration
9  declaree = identifier | declaree "[" integer "]" | declaree "=" expression
10 type = ("int" | "bool" | "string")
11
12 assignmentStatement = address "=" expression
13 address = identifier | address "[" expression "]"
14
15 expression = listExpression | expression ("+" | "-" | "or") term | term
16 listExpression = "{" "}" | "{" expression {"," expression} "}"
17 term = term ("*" | "/" | "%" | "and") factor | factor
18 factor = "(" expression ")" | relational | address | constant
19 relational = expression comparator expression
20 comparator = "<" | "<=" | "==" | ">=" | ">"
21 constant = integer | boolean | string
22
23 ioStatement = readStatement | writeStatement
24 readStatement = "read" "(" address ")"
25 writeStatement = "write" "(" expression ")"
26
27 ifStatement = "if" "(" expression ")" "then" "{" block "}" ["else" "{" block "}"]
28
29 loopStatement = whileStatement | forStatement
30 forStatement = "for" "(" (declarationStatement | assignmentStatement) ";" expression ";"
assignmentStatement ")" "do" "{" block "}"
31 whileStatement = "while" "(" expression ")" "do" "{" block "}"
32
33 integer = "0" | ["+" | "-"] nonzero {digit}
34 boolean = "true" | "false"
35 string = "" {character} ""
36
37 identifier = (letter | "_") {character}
38 character = letter | digit | "_"
39 letter = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
40 digit = "0" | nonzero
41 nonzero = "1" | "2" | ... | "9"

```