==========Lexic.txt ==========

Alphabet:

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

b. Underline character '_';

c. Decimal digits (0-9);

Lexic:

a. Special symbols

-operators +, -, *, /, ==, <, >, <=, >=, =, !=

-separators [], {}, (), ;, space, "", $

-reserved words: Integer, String, Char, for, return, in, out, if, elif, else, ArrayList, while

b. Identifiers

Sequence of letters, digits or "_", with the condition the first character is a letter.

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

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

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

c. Constants

1. integer

<intconst> ::= <unsigned> | -<unsigned> | 0

<unsigned> ::= <nonzerodigit> | <nonzerodigit><digitseq>

<nonzerodigit> ::= 1 | 2 | 3 | ... | 9

<digitseq> ::= <digit> | <digit><digitseq>

<digit> ::= 0 | <nonzerodigit>

2.character

<charconst> ::= "<letter>" | "<digit>"

3.string

<stringconst>::= "" | "<string>"

<string> ::= <char> | <string><char>

<char> ::= <letter> | <digit> | _ | " "

==========syntax.in==========

<program> ::= $<compoundstmt>$

<compoundstmt> ::= <stmtlist>

<stmtlist> ::= <stmt>; | <stmt>;<stmtlist>

<stmt> ::= <assignstmt> | <declrstmt> | <iostmt> | <ifstmt> | <forstmt> | <whilestmt> | <returnstmt>

<exp> ::= <exp> + <term> | <exp> - <term> | <term>

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

<factor> ::= (<exp>) | <identifier> | <intconst>

<arrayint> ::= <intconst> | <arrayint>,<intconst>

<arraystring> ::= <stringconst> | <arraystring>,<stringconst>

<assignstmt> ::= <identifier>=<exp> | <identifier>={<arrayint>} | <identifier>={<arraystring>}

<declrstmt> ::= <type> <identifierlist>

<identifierlist> ::= <identifier> | <identifier>, <identifierlist>

<type> ::= <simpletype> | <arraytype>

<simpletype> ::= Integer | String | Char

<arraytype> ::= ArrayList[<simpletype>]

<iostmt> ::= in(<identifier>) | out(<identifier>) | out(<stringconst>)

<ifstmt> ::= if(<condition>){<compoundstmt>} |
if(<condition>){<compoundstmt>}elif(<condition>){<compoundstmt>} |
if(<condition>){<compoundstmt>}else{<compoundstmt>} |
if(<condition>){<compoundstmt>}elif(<condition>){<compoundstmt>}else{<compoundstmt>}

<condition> ::= <exp><relation><exp>

<relation> ::= < | > | == | != | <= | >=

<forstmt> ::= for<fordeclr>{<compoundstmt>}

<fordeclr> ::= (<assignmentstmt>; <condition>; <assignmentstmt>)

<whilestmt> ::= while(<condition>){<compoundstmt>}

<returnstmt> ::= return <exp>

==========token.in==========

+

-

*

%

/

=

==

!=

<

>

<=

>=

{

}

(

)

[

]

"

_

in

out

return

for

while

if

elif

else

Char

Integer

String

ArrayList