Lexic.txt

Alphabet:

-upper and lower case letters of the English alphabet <letter>

-@ character

-decimal digits (0-9) <digit>

-operators <operator>

-separator <separator>

Identifiers:

-any combination of letters or digits that start with the @ character but the first character after the @ character needs to be a letter

Constants:

-integer:

<non-zero digit> ::= 1| ... | 9

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

<sign> ::= +|-

<unsigned integer> ::= <non-zero digit> | <unsigned integer> <digit>

<signed integer> ::= 0 | <unsigned integer> | <sign> <unsigned integer>

-character:

<character literal> := digit | letter

<character const> := "'" {character literal} "'"

-string:

<char> := letter | digit

<string> := "{character literal}"

Special symbols, representing:

-arithmetic operators: + - % * /

-relational operators: = < <= > >= == ? :=

-separators: () {} [] : ; space ?

-reserved words: int char string array execute else if while of W

R

token.in

Reserved words:

int

char

string

array

const

execute

else if

while

for of

W

R

Operators:

+

- *

%

/

!

!=

==

+=

-=

/=

*=

<

>

<=

>=

||

&&

?

Separators:

[

]

{

}

(

)

; :

,

"

'

Syntax.in

<type> ::= int | char | string | array

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

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

<identifier>:= @<letter><digit>

<factor> ::= (<expression>) | <identifier>

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

<term> ::= <term><term operator><factor>|<factor>

<expression operator>::=+|-

<expression> ::= <expression><expression operator><term>|<term>|<ternary expression>

<condition> ::= <expression><relational operator><expression>

<ternary expression>::= <condition>?<expression>:<expression>

<decl stmt>::=<type> <identifier>; | <type><identifier>=<expression>;

<assign stmt>::= <identifier>=<expression>;

<iostatement>::= R(<identifier>)|W(<identifier>)

<if statement>::=if(<condition>) execute {statement_list}|if(<condition>) execute {statement_list} else execute {statement-list}

<while statement>::=while(<condition>) execute {statement-list}

<relational operator>::= "<" | "<=" | "=" | "<>" | ">=" | ">"

<statement>::= <assign stmt>|<iostatement>|<if statement>|<while statement>|<for statement>

<statement-list>::=<statement>|<statement-list><statement>

<program>::= null|<statement-list>

Program 1 (p1): Max of 3 Numbers

```
int @a, @b, @c, @max;

R(@a);

R(@b);

R(@c);

@max = (@a > @b) ? @a : @b;

@max = (@max > @c) ? @max : @c;

W(@max);
```

Program 2 (p2): GCD of 2 Numbers

```
int @a, @b;
R(@a); R(@b);
while (@b != 0) execute {
int @temp = @b;
   @b = @a % @b;
   @a = @temp;
}
W(@a);
```

Program 3 (p3): Sum of n Numbers and Max/Min of n Numbers

```
int @n, @i, @sum, @num, @max, @min;
R(@n);
@sum = 0;
R(@num);
@max = @num;
@min = @num; @sum
+= @num;
for (@i = 1; @i < @n; @i += 1) execute {
   R(@num);
   @sum += @num;
   @max = (@num > @max) ? @num : @max;
   @min = (@num < @min) ? @num : @min;
}
W(@sum);
W(@max);
W(@min);
```

Erroneous Program (p1err)

```
int @1a, @b, @c; // Error: digit before letter in identifier

R(@a);

R(@b);

R(@c);

@max == (@a > @b) ? @a : @b;   // Error: Used == instead of =

@max = (@max >= @c) ? max : @c; // Error: undeclared variable 'max'

Write(@max);
```