Lab 1a updated

p1: max of 3 numbers

```
Func {
	number a;
	number b;
	number c;
	number maxi;

	read(a);
	read(b);
	read(c);
	if ( a > b and a > c ) {
		maxi = a;
	}

	if ( b > a and b > c ) {
		maxi = b;
	}

	if ( c > a and c > b ) {
		maxi = c;
	}

	text message = 'max is: ';

	write(message, maxi);
}

p1err:
```

```
Func  {
        //error declaring #1a
        number #1a;
        number b;
        number c;
        number maxi;

        read(#1a);
        read(b);
        read(c);
        if ( #1a > b and #1a > c ) {
                maxi = #1a;
        }

        if ( b > #1a and b > c ) {
                maxi = b;
        }

        if ( c > #1a and c > b ) {
                maxi = c;
        }

        text message = 'max is: ';

        //write without ","
        write(message maxi);
}
```

p2: compute gcd of 2 numbers

```
Func  {
```

```
	number a;
	number b;

	read(a);
	read(b);

	while (b not 0) {
		number r = a % b;
		a = b;
		b = r;
	}

	write("gcd: ", a);
}

p3: compute the sum of n numbers
Func  {
	number array arr;
	number lengthArray;
	read(lengthArray);
	number sum = 0;
	number i;
	for (i=0; i<lengthArray; i=i+1) {
		read(arr[i]);
		sum = sum + arr[i];
	}

	text message = 'sum of numbers is: ';
	write(message, sum);
}
```

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, representing:

- operators + - * / = < <= == >= % and or not

- separators [ ] { } ( ) ; , space

- reserved words:

array text number const if while for read write Func

b.identifiers

-a sequence of letters and digits, such that the first character is a letter (a, abc, a1c, etc):

<identifier> ::= <letter> | letter{letter}{digit}

<letter> ::= <capital_letter> | <small_letter>

<capital_letter> ::= A | B | ... | Z

<small_letter> ::= a | b | ... | z

<digit> ::= 0 | <nonzerodigit>

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

c.constants

1.integer - rule: doesnt allow +-0, 001, +01, etc

<integer> ::= 0| <nr> | <sign><nr>

<sign> ::= + | -

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

<nr> ::= <nonzerodigit> | <nonzerodigit><digitseq>

2.character

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

3.string

<string> ::= <character> | <character><string>

CONSTANT=integer|character|string

Syntax.in

The words - predefined tokens are specified between " and ":

Sintactical rules: (file Syntax.in)

<program> ::= "Func" <cmpdstmt>

<simpletype> ::= "number" | "text"

<arraydecl> ::= <simpletype> "array" IDENTIFIER

<type> ::= <simpletype>|<arraydecl>

<declaration> ::= <type> " " IDENTIFIER ";"

<cmpdstmt> ::= "{" <stmtlist> "}"

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

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

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

<assignstmt> ::= IDENTIFIER "=" <expression>

<expression> ::= <expression> "+" <term> | <term>

<term> ::= <term> "*" <factor> | <factor>

<factor> ::= "(" <expression> ")" | IDENTIFIER

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

<ifstmt> ::= "if" "(" <condition> ")" "{" <stmt> "}" ["else" "{" <stmt> "}"]

<whilestmt> ::= "while" "(" <condition> ")" "{" <stmt> "}"

<forstmt> ::= "for" "(" <assignstmt> ";" <condition> ";" <assignstmt> ")" "{" <stmt> "}"

<iostmt> ::= "read" "(" IDENTIFIER ")" | "write" "(" IDENTIFIER|CONSTANT ")" ";"

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

<RELATION> ::= "<" | "<=" | "=" | "<>" | ">=" | ">"

Token.in

Token List

+

-

*

/

=

<

<=

==

>=

%

and

or

not

[

]

{

}

(

)

;

,

space

array

text

number

const

if

while

for

read

write

Func