Lab1_b

1) Alphabet:

a. [A-Za-z]

b. [0-9]

c. Underscore ('_')

2) Lexic:

a. Special symbols, representing:

-operators: + - * / % = < > == >= <= != and or ! []

-separators: {} () , . : ; <space> <newline>

-reserved words: read, write, begin, end, string, int, list,

for, verify, while, elverify, else, declare, as, boolean, character

b. identifiers:

-a sequence of letters and digits,

such that the first character is a letter; the rule is:

identifier = letter({letter|digit|underscore})

letter = "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

digit = "0" | non_zero_digit

non_zero_digit = "1" | ... | "9"

underscore = "_"

c. constants

1.integer - rule: -0 or 01 or other stuff derived from these are not accepted

integer = "0" | [("+" | "-")] non_zero_digit{digit}

2.character

character_part = letter | digit

character = "'"character_part"'"

3.string

string = "{character_part}"

4. boolean

boolean = “True” | “False”

const = integer | character | string | boolean

Token:

(

)

[

]

{

}

and

or

+

-

*

%

/

=

<

<=

>

>=

==

!

!=

.

,

;

:

<space>

<newline>

list

begin

end

read

write

int

string

boolean

verify

elverify

else

declare

as

for

while

Syntax.in:

program = "begin" ";" decllist cmpdstmt end ";"

decllist = {declaration}

declaration = "declare as " type ":" {IDENTIFIER} ";"

type = type1 | arraydecl

type1 = "boolean" | "character" | "int" | “string”

arraydecl = "list" "[" nr "]" "OF" type1

cmpdstmt = stmtlist

stmtlist = {stmt}

stmt = simplstmt | structstmt

simplstmt = assignstmt | iostmt

assignstmt = IDENTIFIER "=" expression ";" | indexidentif "=" expression ";"

expression = expression ("+" | "-") term | term

term = term ("*" | "/" | "%") factor | factor

factor = "(" expression ")" | IDENTIFIER | int | indexidentif | const

indexidentif = IDENTIFIER "[" int "]"

iostmt = readstmt | writestmt

readstmt = "read" "(" IDENTIFIER ")" ";"

writestmt = "write" "(" expression ")" ";"

structstmt = ifstmt | whilestmt | forstmt

ifstmt = "verify" "(" condition ")" "{" cmpdstmt "}" {"elverify" "(" condition ")" "{" cmpdstmt "}" } ["else" "{" cmpdstmt "}"]

whilestmt = "while" "(" condition ")" "{" cmpdstmt "}"

forstmt = "for" forhead "{" cmpdstmt "}"

forhead = "(" "int" assignstmt ";" condition ";" assignstmt ")"

condition = cond | cond logical cond | "!" cond

cond = expression RELATION expression

RELATION = "<" | "<=" | "==" | "!=" | ">=" | ">"

logical = "and" | "or"

Lab1_a_updated:

1. compute the max of 3 nrs:

```
begin;

        declare as int: a, b, c;
         read(a);
         read(b);
         read(c);
         verify(a>=b and b>=c)
                {write(a);}
         elverify(b>=a and a>=c)
                {write(b);}
         elverify(c>=a and a>=b)
                {write(c);}

end;
```

1a. error:

```
begin;

         declare as int: 1a, 2b, c;
         declare as string: "aa;
         read(a);
```

```
        read(b);
        read(c);
        verify(a>=b and b>=c)
                {write(a);}
        elverify(b>=a and a>=c)
                {write(b);}
        elverify(c>=a and a>=b)
                 {write(c);}

end;
```

2. compute the sum of n numbers:

```
begin;

        declare as int: a, sum=0, n;
        read(n);
        while(n>0)
        {
                read(a);
                sum = sum + a;
              n = n - 1;
        }
        write(sum)

end;
```

3. compute the gcd of 2 nrs

```
begin;

        declare as int: a,b;
        read(a);
        read(b);
        verify(a == 0)
                {write(b);}
        verify(b == 0)
                {write(a);}
        verify(a==b)
                {write(a);}
        while(a!=b)
        {
                verify(a>b)
                        {a = a - b;}
                else
                        {b = b - a;}
        }
        write(a);

end;
```