Lexic

a. Special symbols, representing:

- Operators + - * / % < <= = \= => > := and or not
- Separators [ ] { } , . space
- Reserved words: declare read int string boolean list for while function if otherwise then print
- Comments ~

b. Identifiers: a sequence of letters with no digits

- identifier ::= letter { letter }
- letter ::= “A” | “B” | … | “Z” | “a” | “b” | … | “z” c. Constants:
- Integer:

intconst ::= [“+” | “-“] nzDigit { “0” | nzDigit }

digit ::= “0” | “1” | … | “9”

nzDigit ::= “1” | … | “9”

- String:

stringconst ::= “ \” “ { letter | digit | specialSymbol } ” \” “

- Boolean:

booleanconst ::= “true” | “false”

Tokens

int string boolean declare read int string boolean list for while function if otherwise then print ~ + - * / % < <= = \= => > [ ] { } , .

Syntax

Program ::= Stmt { Stmt }

Stmt ::= (DeclareStmt | AssignStmt | PrintStmt | IfStmt | WhileStmt | ForStmt | ReadStmt) “.”

DeclareStmt ::= “declare” type (identifier [“[“ intconst “]”] | AssignStmt)

Type ::= “int” | “string” | “boolean”

AssignStmt ::= identifier “:=” term

Term ::= factor [operator (factor | term)]

Operator ::= “+” | “-“ | “*“ | “/“ | “%“

Factor ::= identifier | intconst | stringconst | booleanconst

IfStmt ::= “if” Condition “then” Stmt { Stmt } [“otherwise” Stmt { Stmt }]

Condition ::= term relOperator term

RelOperator ::= “<” | “> “ | “=” | “<=” | “=>” | “\=”

WhileStmt ::= “while” Condition “execute” Stmt { Stmt }

ForStmt ::= “for” identifier “:<” (intconst | term) “,” (intconst | term) “,” intconst “>” Stmt { Stmt }

PrintStmt ::= “print” term

ReadStmt ::= “read” type identifier

LAB 1

p1

```
declare int a, b, c, max.  read
int a, b, c.  max :=
a.

if a < b then
{
        max := b.
        if b < c then max := c.
}
otherwise  if a < c then max
        := c.   print
max.
```

p2

```
declare int nr.
declare boolean ok := true.  read int
nr.

if nr < 2 then ok := false.

for i: <2, nr / 2, 1>  if nr % i = 0 then ok
        := false.
print ok.
```

p3

```
declare int n, sum := 0.  declare list
int numbers[n].
read int n.

for i: <0, n - 1, 1>  read numbers[i].
        sum := sum + numbers[i].
print sum.
```

p1err

declare str “mystring .   ~didn’t close quotation marks declare int
n3.   ~identifiers can contain only letters