**program** :=   
 statement+

**statement** :=

assignmentStatement

conditionalStatement

loopStatement

callStatement

listStatement

loopOnlyStatements

returnStatement

commentStatement

**assignmentStatement** :=

*set* identifier *equal to* expression.

**conditionalStatement** :=

*if* booleanExpression *then*:

statement+

[*otherwise*:

statement+]

**loopStatement** :=

*loop while* booleanExpression:

statement+

*loop* [identifier] *from* numberExpression *to* numberExpression [by numberExpression]

statement+

*loop over* identifier [*into* identifier [*named* identifier]]:

statement+

**callStatement** :=

*call* identifier [arguments].

**loopOnlyStatements** :=

*next*.

*exit loop*.

**returnStatement** :=

*return* [expression].

**Identifier** :=

\w+( \w+)\*

no part of the identifier is allowed to be a reserved key word (italics)

**expression** :=

numberExpression

booleanExpression

stringExpression

callExpression

functionExpression

newListExpression

listExpression

**numberExpression** :=

number

numberArthimeticExpression

**number** :=

(-)?(\.\d+|\d+(\.\d\*)?)

**numberArithmeticExpression**

number *plus* number

number *minus* number

number (*multiplied*|*times*) number

number *divided by* number

number (*modulo*|*mod*) *by* number

(*negate*|*negative*) number

**booleanExpression**

boolean

booleanArithmeticExpression

**boolean** :=

*true*

*false*

**booleanArithmeticExpression** :=

*is* number [*not*] *equal to* number

*is* number *greater than* number

*is* number *greater than or equal to* number

*is* number *less than* number

*is* number *less than or equal to* number

*cis* string *greater than* string

*is* string *greater than or equal to* string

*is* string *less than* string

*is* string *less than or equal to* string

*is* boolean [*not*] *equal to* boolean

boolean *and* boolean

boolean *or* boolean

*not* boolean

boolean *exclusive or* boolean

**stringExpression** :=

string

stringArithmeticExpression

**string** :=

“.\*?”

**stringArithmeticExpression** :=

string *append* string

**callFunctionExpression** :=

*call* identifier [arguments]

**functionExpression :=**

*new function* [arguments]:

statement+

**arguments :=**

argument\*

**argument :=**

*with* identifier

**newListExpression** :=

new list

**listStatement** :=

*push* expression *onto* identifier [*named* stringExpression].

*shift* expression *onto* identifier [*named* stringExpression].

*insert* expression *into* identifier *at* expression.

*remove* *from* identifier *at* expression.

**listExpression** :=

*pop* *from* identifier

*unshift* *from* identifier

*length of* identifier

*element from* identifier *at* expression

**commentStatement** :=

#.\*

BuildInFunctionStatement :=

*print* expression.

BuildInFunctionExpression :=

*read* *line*

*type* *of* identifier