MEG Grammar

<stmt> ::= <expr> | <for> | <if> | <show>

<expr> ::= <int_to_x> | <str_to_x> | <bool_to_x>

<int_to_x> ::= <int_to_int> | <int_to_expr>
<int_to_int> ::= _int_ <var> -> <num>.
<int_to_expr> ::= _int_ <var> -> <num> <int_op> <rest>.
<rest> ::= <num>. | <num> <int_op> <rest>.
<int_op> ::= + | - | * | / | %
<num> ::= <num1> | -<num1>
<num1> ::= <num2> | <num2><num1>
<num2> ::= 0 | … | 9
<var> ::= <int_var>

<str_to_x> ::= _string_ <var2> -> (<var>).
<var2> ::= <alp> | <alp><all_var>
<all_var> ::= <all_var2> | <all_var><all_var2>
<all_var> ::= *any number, letter, or symbol*

<bool_to_x> ::= <bool_to_bool> | <bool_to_expr>
<bool_to_bool> ::= _bool_ <var2> -> <bools>.
<bools> ::= true | false | ! true | ! false | <var2> | ! <var2>
<bool_to_expr> ::= _bool_ <var2> -> <bools> <bool_op> <rest>.
<rest> ::= <bools> | <bools> <bool_op> <bools>.
<bool_op> ::= | | &

<int_var> ::= <alp> | <alp><int_var>
<alp> ::= a | … | z | A … | Z
<for> ::= for <var> -> <num> <num> <num>. [<stmt>] | for <var> -> <int_var> <num> <num>. [<stmt>] | for <var> -> <num> <int_var> <num>. [<stmt>] | for <var> -> <num> <num> <int_var>. [<stmt>] | for <var> -> <int_var> <int_var> <num>. [<stmt>] | for <var> -> <int_var> <num> <int_var>. [<stmt>] | for <var> -> <num> <int_var> <int_var>. [<stmt>] | for <var> -> <int_var> <int_var> <int_var>. [<stmt>]

<show> ::= show <var2>.

<if> ::= ifeq <var> <num>. [<stmt>] | ifneq <var> <num>. [<stmt>] | ifgt <var> <num>. [<stmt>] | iflt <var> <num>. [<stmt>] | iflteq <var> <num>. [<stmt>] | ifgteq <var> <num>. [<stmt>]