

CSC 372 Project 2 Grammar

integers

$\langle \text{digit} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\langle \text{integer} \rangle ::= \langle \text{digit} \rangle \mid \langle \text{integer} \rangle \langle \text{digit} \rangle$

variables

$\langle \text{letter} \rangle ::= A \mid B \mid C \mid D \mid E \mid F \mid G \mid H \mid I \mid J \mid K \mid L \mid M \mid N \mid O \mid P \mid Q \mid R \mid S \mid T \mid U \mid V \mid W \mid X \mid Y \mid Z \mid a \mid b \mid c \mid d \mid e \mid f \mid g \mid h \mid i \mid j \mid k \mid l \mid m \mid n \mid o \mid p \mid q \mid r \mid s \mid t \mid u \mid v \mid w \mid x \mid y \mid z$

$\langle \text{variable} \rangle ::= \langle \text{letter} \rangle \mid \langle \text{variable} \rangle \langle \text{letter} \rangle$

variable assignment

$\langle \text{var_asgmt} \rangle ::= \text{integer } \langle \text{variable} \rangle \text{ assign } \langle \text{int_expr} \rangle \mid \text{boolean } \langle \text{variable} \rangle \text{ assign } \langle \text{bool_expr} \rangle \mid \langle \text{variable} \rangle \text{ assign } \langle \text{expr} \rangle$

$\langle \text{expr} \rangle ::= \langle \text{int_expr} \rangle \mid \langle \text{bool_expr} \rangle$

integer expression

$\langle \text{int_expr} \rangle ::= \langle \text{mult_expr} \rangle \mid \langle \text{int_expr} \rangle \text{ sub } \langle \text{mult_expr} \rangle \mid \langle \text{int_expr} \rangle \text{ add } \langle \text{mult_expr} \rangle$

$\langle \text{mult_expr} \rangle ::= \langle \text{negate_expr} \rangle \mid \langle \text{mult_expr} \rangle \text{ mult } \langle \text{negate_expr} \rangle \mid \langle \text{mult_expr} \rangle \text{ div } \langle \text{negate_expr} \rangle \mid \langle \text{mult_expr} \rangle \text{ mod } \langle \text{negate_expr} \rangle$

$\langle \text{negate_expr} \rangle ::= \langle \text{int_root} \rangle \mid \text{negate } \langle \text{int_root} \rangle$

$\langle \text{int_root} \rangle ::= \langle \text{integer} \rangle \mid \langle \text{variable} \rangle \mid \langle \text{cmd_arg_var} \rangle$

booleans

$\langle \text{boolean} \rangle ::= \text{true} \mid \text{false}$

boolean expression

$\langle \text{bool_expr} \rangle ::= \langle \text{and_expr} \rangle \mid \langle \text{bool_expr} \rangle \text{ or } \langle \text{and_expr} \rangle$

$\langle \text{and_expr} \rangle ::= \langle \text{not_expr} \rangle \mid \langle \text{and_expr} \rangle \text{ and } \langle \text{not_expr} \rangle$

$\langle \text{not_expr} \rangle ::= \langle \text{bool_root} \rangle \mid \text{not } \langle \text{bool_root} \rangle \mid \langle \text{int_expr} \rangle \langle \text{comparator} \rangle \langle \text{int_expr} \rangle$

$\langle \text{comparator} \rangle ::= \text{lt} \mid \text{gt} \mid \text{lte} \mid \text{gte} \mid \text{equal} \mid \text{diff}$

$\langle \text{bool_root} \rangle ::= \langle \text{boolean} \rangle \mid \langle \text{variable} \rangle$

conditionals

```
<conditional> ::=      if <bool_expr>
                        <mult_cond_loop>
                        end |
                        if <bool_expr>
                        <mult_cond_loop>
                        else
                        <mult_cond_loop>
                        end
<mult_cond_loop> ::=    <root_cond_loop> |
                        <mul_cond_loop>
                        <root_cond_loop>
<root_cond_loop> ::= <var_asgmt> | <print> | <loop> | <conditional>
```

loops

```
<loop> ::=      while <bool_expr>
                <mult_cond_loop>
                end
```

printing to output

```
<print> ::= printout <print_root> | print <print_root>
<print_root> ::= <int_expr> | <bool_expr> | <string_literal>
```

command line arguments

```
<cmd_arg_define> ::= use <integer> cmd args
<cmd_arg_var> ::= arg<integer>
```

string literals

```
<string_literal> ::= "<string>"
<string> ::= <character> | <string><character>
<character> ::= Any ASCII character
```

comments

```
<comment> ::= #<comment_line>
<comment_line> ::= <comment_char> | <comment_line> <comment_char>
<comment_char> ::= Any ASCII character except for double quote character
```