Eric Mendoza, Luke Ramirez

LEMR Language

**Primitives**:

<char> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |

Y | Z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ` | ~ | ! | @ | # | $ | % | ^ | & | \* | ( | ) | - | \_ |

+ | = | { | } | [ | ] | \ | \| | : | ; | < | \’ | \” | \n | (space) | < | , | > | . | / | ?

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<int> ::= <digit> | <int><digit>  
<boolean> true | false

<primitive> ::= <char> | <boolean> | <int>

**Type Annotations**:

<type> ::= int | bool | char | str

**Strings**:  
<string> ::= <char> | <string><char>

<stringLiteral> ::= “<string>”   
<charLiteral> ::= ‘<char>’

**Variable Assignment**:

<var> ::= <type> <string> = <primitive>  
  
**Integer Expressions**:

<iExpr> ::= <iExpr> + <multExpr> | <iExpr> - <multExpr> | <multExpr>

<multExpr> ::= <multExpr> \* <negExpr> | <multExpr> div <negExpr> |

<multExpr> mod <negExpr> | <negExpr>

<negExpr> ::= <iRoot> | -<iRoot>

<iRoot> ::= <integer> | (<iExpr>) | <var>

**Boolean Expressions**:

<orExpr> ::= <orExpr> \| <andExpr> | <andExpr>

<andExpr> ::= <andExpr> & <compExpr> | <compExpr>

<compExpr> ::= <compExpr> < <notExpr> | <compExpr> > <notExpr> |

<compExpr> <= <notExpr> | <compExpr> >= <notExpr> |

<compExpr> = <notExpr> | <notExpr>

<notExpr> ::= <bRoot> | !<bRoot>

<bRoot> ::= <boolean> | (<orExpr>) | <var>

**Conditionals**:

<conditional> ::= if (<orExpr>) {<expression>}

**Loops**:

<loop> ::= while(<orExpr>) {<expression>} | fori(<string> + <int>, <int>, <int>)

{<expression>} | fori(<string> - <int>, <int>, <int>) {<expression>}

**Printing**:

<print> ::= >{<string>} | >>{<string>}

**Comments**:

<comment> ::= //<string> | /\*<string>\*/

**Arrays**:

<intList> ::= <int> | <intList>, <int>

<intArray> ::= [<intList>]

**Command Line Arguments**:

<args> ::= args(<intArray>)