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> | null

Strings:

<string> ::= <char> | <string><char>

<stringLiteral> ::= “<string>”

<charLiteral> ::= ‘<char>’

Variable Assignment:

<varAssignment> ::= <var> = <primitive> | <var> = <string> | <var> = <iExpr> | <var> = <orExpr>

<var> ::= <string>

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>} | if (<orExpr>) {<expression>} else {<expression>} |
if (<orExpr>) {<expression>} <elif> | if (<orExpr>) {<expression>} <elif> else
{<expression>} | <orExpr> ? <expression> : <expression>

<elif> ::= elif (<orExpr>) {<expression>} | <elif> elif (<orExpr>) {<expression>}

Loops:

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

Printing:

<print> ::= >{<var>} | >{<stringLiteral>} | >{<charLiteral>} | >>{<var>} | >>{<stringLiteral>} |
>>{<charLiteral>}

Comments:

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

Command Line Arguments:

*Implemented as a buffer, that the user reads in from, when empty returns null

<CLargs> ::= <var> = getArg()

Expressions:

<expression> ::= <varAssignment> | <print> | <loop> | <conditional> |

<expression>\n<varAssignment> | <expression>\n<print> | <expression>\n<loop> |

<expression>\n<conditional>