|  |  |
| --- | --- |
| **Group Name: Aquigastro** | **Section: T- 1L** |
| **Member 1: Aquino, Robina Rhamz** | **Member 3: Gragas, George** |
| **Member 2: Castro, Christine Marie** | **Member 4: N/A** |

**LOLCODE GRAMMAR**

Use angle brackets (<,>) to denote abstractions. Type lexemes that have been defined in Project Requirement 01 using lowercase letters. If the lexemes have not yet been defined, add the newly defined lexemes at the last section of this document.

|  |  |  |
| --- | --- | --- |
| **LHS** | **::=** | **RHS** |
| <program> | ::= | hai <linebreak>  <stmt> <linebreak>  kthxbye |
| <stmt> | ::= | <stmt> <linebreak> <stmt2> | <stmt2>  <stmt2> <linebreak> <stmt> |  <stmt2> <inline\_comments> |  <stmt2> |
| <stmt2> |  | <print> |  <multi\_comments> |  <variable\_assignment>|  <input> |  <cond\_stmt> |  <expr> |  <switch\_stmnt> |  <loop\_stmt> |  <typecast\_stmt> |  <recast\_stmt> |  <str\_concat> |
| <print> | ::= | visible <print\_args> |
| <print\_args> | ::= | <print\_args> <print\_args2> |  <print\_args2> |
| <print\_args2> | ::= | varident |  <expr> <literal> |
| <input> | ::= | gimmeh varident |
| <literal> | ::= | numbr |  numbar |  yarn |  troof |
| <number\_literal> | ::= | numbr |  numbar |
| <linebreak> | ::= | newline |  softbreak |
| <multi\_comments> | ::= | obtw <comment> <linebreak>  tldr |  obtw <comment> <linebreak>  <comment> tldr |
| <inline\_comments> | ::= | btw <literal> |
| <variable\_assignment> | ::= | i has a varident |  i has a varident itz <literal> |  i has a varident itz varident |  i has a varident itz <expr> |  varident R <literal> |  varident R varident |  varident R <expr> |
| <expr> | ::= | not <expr> |  <binary\_exp> |  <infinite\_arity\_expr> |
| <not\_infinite\_expr> |  | not <expr> |  <binary\_exp> |
| <binary\_exp> | ::= | <binary\_math\_operator> <binary\_exp> an <binary\_exp> |  <literal> |  <binary\_boolean\_operator> <binary\_exp> an <binary\_exp> |  varident  X  Redundant so use the below  <binary\_math\_operator> |  <literal> |  <binary\_boolean\_operator> |  varident |
| <binary\_math\_operator> | ::= | sum of <binary\_exp> an <binary\_exp> |  diff of <binary\_exp> an <binary\_exp> |  produkt of <binary\_exp> an <binary\_exp> |  quoshunt of <binary\_exp> an <binary\_exp> |  mod of <binary\_exp> an <binary\_exp> |  biggr of <binary\_exp> an <binary\_exp> |  smallr of <binary\_exp> an <binary\_exp> |  <binary\_exp>  x  does binary\_exp need? Look at below table for fixed  sum of <binary\_exp> an <binary\_exp> |  diff of <binary\_exp> an <binary\_exp> |  produkt of <binary\_exp> an <binary\_exp> |  quoshunt of <binary\_exp> an <binary\_exp> |  mod of <binary\_exp> an <binary\_exp> |  biggr of <binary\_exp> an <binary\_exp> |  smallr of <binary\_exp> an <binary\_exp> |
| <binary\_boolean\_operator> |  | both of <binary\_exp> an <binary\_exp> |  either of <binary\_exp> an <binary\_exp> |  won of <binary\_exp> an <binary\_exp>  X  Changed at below |
| <comparison\_operators> |  | both saem <number\_literal> an biggr of <number\_literal> an <number\_literal> |  both saem <number\_literal> an smallr of <number\_literal> an <number\_literal> |  diffrint <number\_literal> an smallr of <number\_literal> an <number\_literal> |  diffrint <number\_literal> an biggr of <number\_literal> an <number\_literal> |  number literal can be expression, fuck it’s longer |
| <infinite\_arity\_expr> | ::= | all of <literal> an <literal> <infite\_arity\_expr\_end1>  x  all of <not\_infinite\_expr> an <not\_infinite\_expr> <infinite\_arity\_expr\_end1> |
| <infite\_arity\_expr\_end1> | ::= | <infite\_arity\_expr\_end1> <infite\_arity\_expr\_end2> |  <infite\_arity\_expr\_end2> |
| <infite\_arity\_expr\_end2> | ::= | mkay |  an <literal> |
| <cond\_stmt> | ::= | <expr> <linebreak> o rly? <linebreak> ya rly <linebreak>  <stmt> <linebreak> oic|  <expr> <linebreak> o rly? <linebreak> ya rly <linebreak>  <stmt> <linebreak> no wai <linebreak>  <stmt> <linebreak> oic|  <expr> <linebreak> o rly? <linebreak> ya rly <linebreak>  <stmt> <linebreak> <mebbe\_stmt> |
| <mebbe\_stmt> | ::= | mebbe <expr> <linebreak> <stmt> <linebreak> <mebbe\_stmt>|  mebbe <expr> <linebreak> <stmt> <linebreak> oic|  mebbe <expr> <linebreak> <stmt> <linebreak>  no wai <linebreak> <stmt> <linebreak> oic |
| <switch\_stmt> | ::= | wtf? <linebreak> <omg\_stmt> <linebreak> oic |
| <omg\_stmt> | ::= | omg <literal> <linebreak> <stmt> |  omg <literal> <linebreak> <stmt> gtfo|  omg <literal> <linebreak> <stmt> <linebreak> <omg\_stmt>|  omg <literal> <linebreak> <stmt> omgwtf <linebreak> <stmt> |
| <loop\_stmt> | ::= | im in yr loopident <loop\_operator> yr varident <loop\_condition> <linebreak> <stmt> <linebreak> im outta yr loopident |
| <loop\_operator> | ::= | uppin | nerfin |
| <loop\_condition> | ::= | til <expr> | wile <expr> |
| <str\_concat> | ::= | smoosh <an\_yarn> an yarn |
| <an\_yarn> | ::= | yarn | <an\_yarn> an yarn |
| <typecast\_stmt> | ::= | maek varident a type |
| <recast\_stmt> | ::= | varident is now a type | varident r maek varident type |

**NEWLY-ADDED LEXEMES**

Put here the definition of the lexemes that have not yet been defined in Project Requirement 01.

|  |  |
| --- | --- |
| **LEXEME** | **Regular Expression** |
| AN | ^an$ |
| NEWLINE | ^\\n$ |
| SOFTBREAK | ^,$ |
| GTFO | ^GTFO$ |

|  |  |  |
| --- | --- | --- |
| **LHS** | **::=** | **RHS** |
| <program> | ::= | “Code Delimiter Start” <linebreak>  <stmt> <linebreak>  “Code Delimiter End” |
|  |  | HAI  VISIBLE A B C D E F  KTHXBYE |
| <stmt> | ::= | <stmt> <linebreak> <stmt2> |  <stmt2> <inline\_comments> |  <stmt2>  X  Stmt2 at first to prevent infinite loop  <stmt2> <linebreak> <stmt> |  <stmt2> <inline\_comments> |  <stmt2> |
| <stmt2> | ::= | <print> |  <multi\_comments> |  <variable\_assignment>|  <input> |  <cond\_stmt> |  <switch\_stmnt> |  <loop\_stmt> |  <typecast\_stmt> |  <recast\_stmt> |  <str\_concat> |  <expr> |  Str concat moved to expr  Typecast stmt moved to expr |
| <output> | ::= | “Output” <output\_args> |
| <output\_args> | ::= | <output\_args> <output\_args2> |  <output\_args2>  X  <output\_args> <expr> | <expr> |
| <output\_args2> | ::= | “Identifiers” |  <expr> |  <literal> |
| <input> | ::= | “Input” “Identifiers” |
| <literal> | ::= | “Numbr Literal” |  “Numbar Literal” |  “Yarn Literal” |  “Troof Literal” |
| <number\_literal> | ::= | “Numbr Literal” |  “Numbar Literal” |
| <linebreak> | ::= | “Newline” | “Softbreak” |
| <multi\_comments> | ::= | “Multiline Comment Delimiter Start” “Comment” <linebreak> “Multiline Comment Delimiter End” |  “Multiline Comment Delimiter Start” “Comment” <linebreak> <comment> “Multiline Comment Delimiter End” |
| <comment> | ::= | <comment> <comment2> |  <comment2> |
| <comment2> | ::= | <literal> <linebreak> |
| <inline\_comments> | ::= | “Inline Comment Delimiter” “Comment” |
| <variable\_assignment> | ::= | “Variable Declaration” “Identifiers” |  “Variable Declaration” “Identifiers” “Variable Declaration Assignment” <literal> |  “Variable Declaration” “Identifiers” “Variable Declaration Assignment” “Identifiers” |  “Variable Declaration” “Identifiers” “Variable Declaration Assignment” <expr> |  “Identifiers” “Variable Assignment” “Identifiers” |  “Identifiers” “Variable Assignment” <literal> |  “Identifiers” “Variable Assignment” <expr> |
| <expr> | ::= | <bool\_expr> |  <infinite\_arity\_expr> |  Added  <str\_concat> |
| <bool\_expr> | ::= | “Not Boolean Operator” <expr> |  <binary\_exp> |
| <binary\_exp> | ::= | <binary\_math\_operator> |  <binary\_bool\_operator> |  <literal> |  “Identifiers” |  <comparison\_operator> |
| <binary\_math\_operator> | ::= | “Math Operator” <binary\_math\_operator> “Expression AND Operator” <binary\_math\_operator>|  “Math Operator” <binary\_math\_operator> “Expression AND Operator” <literal>|  “Math Operator” <binary\_math\_operator> “Expression AND Operator” “Identifiers”|  “Math Operator” <literal> “Expression AND Operator” <binary\_math\_operator>|  “Math Operator” <literal> “Expression AND Operator” <literal>|  “Math Operator” <literal> “Expression AND Operator” “Identifiers”|  “Math Operator” “Identifiers” “Expression AND Operator” <binary\_math\_operator>|  “Math Operator” “Identifiers” “Expression AND Operator” <literal>|  “Math Operator” “Identifiers” “Expression AND Operator” “Identifiers”|  “Comparison Math Operator” <binary\_math\_operator> “Expression AND Operator” <binary\_math\_operator>|  “Comparison Math Operator” <binary\_math\_operator> “Expression AND Operator” <literal>|  “Comparison Math Operator” <binary\_math\_operator> “Expression AND Operator” “Identifiers”|  “Comparison Math Operator” <literal> “Expression AND Operator” <binary\_math\_operator>|  “Comparison Math Operator” <literal> “Expression AND Operator” <literal>|  “Comparison Math Operator” <literal> “Expression AND Operator” “Identifiers”|  “Comparison Math Operator” “Identifiers” “Expression AND Operator” <binary\_math\_operator>|  “Comparison Math Operator” “Identifiers” “Expression AND Operator” <literal>|  “Comparison Math Operator” “Identifiers” “Expression AND Operator” “Identifiers” |
| <binary\_bool\_operator> |  | “Boolean Operator” <expr> AN <expr> |  “Boolean Operator” <expr> AN “Identifiers” |  “Boolean Operator” <expr> AN <literal> |  “Boolean Operator” <literal> AN <expr> |  “Boolean Operator” <literal> AN “Identifiers” |  “Boolean Operator” <literal> AN <literal> |  “Boolean Operator” “Identifiers” AN <expr> |  “Boolean Operator” “Identifiers” AN “Identifiers” |  “Boolean Operator” “Identifiers” AN <literal> |
| <comparison\_operator> |  | “Comparison Operator” <expr> AN <expr> |  “Comparison Operator” <expr1> AN “Comparison Math Operator” <expr1> AN <expr>  \*expr1 must result to same value |
| <infinite\_arity\_expr> | ::= | all of <literal> an <literal> <infite\_arity\_expr\_end1>  x  “Infinite Boolean Operator” <bool\_expr> an <bool\_expr> <infite\_arity\_expr\_end1> |
| <infinite\_arity\_expr\_operand> |  | “Boolean Operator”/“Not Boolean Operator” <infinite\_arity\_expr\_operand> AN <infinite\_arity\_expr\_operand> |  “Boolean Operator”/“Not Boolean Operator” <infinite\_arity\_expr\_operand> AN “Identifiers” |  “Boolean Operator”/“Not Boolean Operator” <infinite\_arity\_expr\_operand> AN <literal> |  “Boolean Operator”/“Not Boolean Operator” <literal> AN <infinite\_arity\_expr\_operand> |  “Boolean Operator”/“Not Boolean Operator” <literal> AN “Identifiers” |  “Boolean Operator”/“Not Boolean Operator” <literal> AN <literal> |  “Boolean Operator”/“Not Boolean Operator” “Identifiers” AN <infinite\_arity\_expr\_operand> |  “Boolean Operator”/“Not Boolean Operator” “Identifiers” AN “Identifiers” |  “Boolean Operator”/“Not Boolean Operator” “Identifiers” AN <literal> |  <literal> |  “Identifiers”  X  “Not Boolean Operator” <infinite\_arity\_expr\_operand> |  <binary\_exp> |
| <infite\_arity\_expr\_end1> | ::= | <infite\_arity\_expr\_end1> <infite\_arity\_expr\_end2> | <infite\_arity\_expr\_end2> |
| <infite\_arity\_expr\_end2> | ::= | mkay | an <expr> |
| <cond\_stmt> | ::= | <expr> <linebreak> "Conditional Statement Delimiter If-Else Start" <linebreak> "Conditional Statement If" <linebreak>  <stmt> <linebreak> "Conditional Statement Delimiter End"|  <expr> <linebreak> "Conditional Statement Delimiter If-Else Start" <linebreak> "Conditional Statement If" <linebreak>  <stmt> <linebreak> "Conditional Statement Else" <linebreak>  <stmt> <linebreak> "Conditional Statement Delimiter End"|  Below removed  <expr> <linebreak> "Conditional Statement Delimiter If-Else Start" <linebreak> "Conditional Statement If" <linebreak>  <stmt> <linebreak> <mebbe\_stmt> |
| <mebbe\_stmnt> | ::= | "Conditional Statement Elif" <expr> <linebreak> <stmt> <linebreak> <mebbe\_stmt>|  "Conditional Statement Elif" <expr> <linebreak> <stmt> <linebreak> "Conditional Statement Delimiter End"|  "Conditional Statement Elif" <expr> <linebreak> <stmt> <linebreak>  "Conditional Statement Else" <linebreak> <stmt> <linebreak> "Conditional Statement Delimiter End" |
| <switch\_stmnt> | ::= | "Conditional Statement Delimiter Switch Start" <linebreak> <omg\_stmt> <linebreak> "Conditional Statement Delimiter End"  became  "Conditional Statement Delimiter Switch Start" <linebreak> <omg\_stmt> |
| <omg\_stmnt> | ::= | "Conditional Statement Switch" <literal> <linebreak> <stmt>|  "Conditional Statement Switch" <literal> <linebreak> <stmt> "Loop Break Operator"|  "Conditional Statement Switch" <literal> <linebreak> <stmt> <linebreak> <omg\_stmt>|  "Conditional Statement Switch" <literal> <linebreak> <stmt> |  added  <linebreak> "Conditional Statement Delimiter End" |
| <switch\_stmt> |  | "Conditional Statement Delimiter Switch Start" <linebreak> <omg\_stmt> |
| <omg\_stmt> |  | <omg\_statement\_end> |  "Conditional Statement Switch" <literal> <linebreak> <stmt> |  "Conditional Statement Switch" <literal> <linebreak> <stmt> <linebreak> "Loop Break Operator" |
| <omg\_stmt\_end> |  | "Conditional Statement Switch Last" <linebreak> <stmt> <linebreak> "Conditional Statement Delimiter End" | "Conditional Statement Delimiter End" |
| <loop\_stmt> | ::= | "Loop Delimiter Start" loopident “Loop Condition” "Arguments Operator" “Identifiers” <loop\_condition> <linebreak> <stmt> <linebreak> "Loop Delimiter End" loopident |
| <loop\_operator> | ::= | "Loop Condition" |
| <loop\_condition> | ::= | til <expr> | wile <expr> |
|  |  |  |
|  |  |  |
| <str\_concat> | ::= | "Concatenation Operator" <expr> <an\_yarn>  X  "Concatenation Operator" <expr> <an\_yarn2> |
| <an\_yarn> | ::= | <an\_yarn2> | <an\_yarn2> <an\_yarn> |
| <an\_yarn2> | ::= | “Expression AND Operator” “Yarn Literal x  “Expression AND Operator” <expr> | <an\_yarn2> |
| <typecast\_stmt> | ::= | “Typecast Start Operator” “Identifiers” “Typecast Middle Operator” "Type Literal" |
| <recast\_stmt> | ::= | “Identifiers” “Casting Operator” <literal> | “Identifiers” “Variable Assignment” “Typecast Start Operator” “Identifiers” <literal> x  “Identifiers” “Casting Operator” <literal> |  “Identifiers” “Variable Assignment” <typecast\_stmt> |