|  |  |  |
| --- | --- | --- |
| **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> |  <expr> |  <switch\_stmnt> |  <loop\_stmt> |  <typecast\_stmt> |  <recast\_stmt> |  <str\_concat> |
| <print> | ::= | “Output” <print\_args> |
| <print\_args> | ::= | <print\_args> <print\_args2> |  <print\_args2> |
| <print\_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> |
| <bool\_expr> | ::= | “Not Boolean Operator” <expr> |  <binary\_exp> |
| <binary\_exp> | ::= | <binary\_math\_operator> |  <binary\_bool\_operator> |  <literal> |  “Identifiers” |
| <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> “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” <expr> an <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” |
| <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"|  <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" |
| <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> "Conditional Statement Switch Last" <linebreak> <stmt> |
| <loop\_stmt> | ::= | "Loop Delimiter Start" loopident <loop\_operator> "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" "Yarn Literal" <an\_yarn> |
| <an\_yarn> | ::= | <an\_yarn2> | <an\_yarn2> <an\_yarn> |
| <an\_yarn2> | ::= | “Expression AND Operator” “Yarn Literal |
| <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> |
| <multiline\_cmt> | ::= | “Multi-line Comment Delimiter Start” <multiline\_cmt2> “Multi-line Comment Delimiter End” |
| <multiline\_cmt2> | ::= | ”Comment” | “Comment” <multiline\_cmt2> |