|  |  |  |  |
| --- | --- | --- | --- |
| Group Name: | Can I Pabebe? | Section: | B-4L |
| Member 1: | FRIAS, Mon Cedrick | Member 3: | MENGUITO, James Paolo |
| Member 2: | MARCOS, Joseph Matthew | Member 4: |  |

LOLCode, like many other programming languages, has different forms of syntax for its various language constructs, and similarly, can be described by a grammar to generate program strings with correct syntax. Fill out the following table by introducing rules, nonterminal variables, and terminal symbols to make a complete grammar for LOLCode. I show an example of some rules for LOLCode, which you may modify to your liking.

Please use **monospace** fonts [preferably Inconsolata, but other good monospace fonts include Consolas (Windows & Google Docs), Monaco (Mac), and Ubuntu Mono (Ubuntu)]. Specify variables as <brief description>.

|  |  |  |
| --- | --- | --- |
| LHS | ::= | RHS |
| <program> | ::= | HAI<linebreak><statement><linebreak>KTHXBYE |
| <linebreak> | ::= | \n | , |
| <statement> | ::= | <comment> | <vardec> | <inputOutput> | <math> | <boolean> | <compare> | <casting> | <flowControl> | <function> | <functionCall> |
| <comment> | ::= | BTW <statement> <linebreak> | OBTW <statement> <linebreak> <statement> TLDR |
| <vardec> | ::= | I HAS A varident | I HAS A varident ITZ <assignrhs> | I HAS A varident <linebreak> <assignR> |
| <assignR> | ::= | varident R <assignrhs> |
| <assignrhs> | ::= | varident | <literal> |
| <inputOutput> | ::= | GIMMEH varident | VISIBLE <printVar> |
| <printVar> | ::= | yarn | yarn varident | varident | varident yarn |
| <literal> | ::= | numbr | numbar | yarn | troof |
| <expression> | ::= | varident | numbr | numbar | yarn | troof | IT |
| <type> | ::= | TROOF | YARN | NUMBR | NUMBAR | NOOB |
| <math> | ::= | SUM OF <expression> AN <expression> | DIFF OF <expression> AN <expression> | PRODUKT OF <expression> AN <expression> | QUOSHUNT OF <expression> AN <expression> | MOD OF <expression> AN <expression> | BIGGR OF <expression> AN <expression> | SMALLR OF <expression> AN <expression> |
| <boolean> | ::= | BOTH OF <expression> AN <expression> | EITHER OF <expression> AN <expression> | WON OF <expression> AN <expression> | NOT <expression> | ALL OF <expression> AN <expression>...MKAY | ANY OF <expression> AN <expression>...MKAY |
| <compare> | ::= | BOTH SAEM <expression> AN <expression> | BOTH SAEM <expression> <math> AN <expression> | <expression>, BOTH SAEM IT AN <math> | DIFFRINT <expression> AN <expression> | DIFFRINT <expression> <math> AN <expression> | <expression>, DIFFRINT IT AN <math> |
| <casting> | ::= | MAEK varident A <type> | varident IS NOW A <type> | varident R MAEK varident A <type> |
| <flowControl> | ::= | <ifThen> | <case> | <loop> |
| <ifThen> | ::= | <compare> <linebreak> O RLY? <linebreak> <if> OIC | <compare> <linebreak> O RLY? <linebreak> <if> <elseif> OIC | <compare> <linebreak> O RLY? <linebreak> <if> <else> OIC | <compare> <linebreak> O RLY? <linebreak> <if> <elseIf> <else> OIC |
| <if> |  | YA RLY <linebreak> <statement> <linebreak> |
| <elseIf> | ::= | MEBBE <compare> <linebreak> <statement> <linebreak> |
| <else> |  | NO WAI <linebreak> <statement> <linebreak> |
| <case> | ::= | varident <linebreak> WTF? <linebreak> <possibleCase> OIC | |
| <possibleCase> | ::= | <caseWithExit> | <caseWithoutExit> | <default> |
| <caseWithExit> | ::= | OMG <literal> <linebreak> <statement> <linebreak> GTFO <linebreak> |
| <caseWithoutExit> | ::= | OMG <literal> <linebreak> <statement> <linebreak> |
| <default> | ::= | OMGWTF <linebreak> <statement> <linebreak> |
| <loop> | ::= | IM IN YR varident <operation> YR varident (TIL|WILE) <> <linebreak> <statement> <linebreak> IM OUTTA YR varident |
| <operation> | ::= | UPPIN | NERFIN |
| <function> | ::= | HOW IZ I varident YR <args> YR <args> ... <linebreak> <statement> <linebreak> IF YOU SAY SO |
| <args> | ::= | varident | <expression> |
| <functionCall> | ::= | I IZ <varident> YR <args> AN YR <args> ...MKAY |

Notes (if you want to clarify anything about your BNF):

* varident stands for identifier; it is used as a terminal symbol here since identifiers were described using regular expressions in the previous requirement and there is no need to make rules for generating them.
* numbr stands for integer literals; regex defined in previous requirement.
* numbar stands for floating-point literals; regex defined in previous requirement.
* yarn stands for string literals; regex defined in previous requirement.
* troof stands for boolean literals; regex defined in previous requirement.

**Address any further questions to me or your lab instructors. Alam nila kung ano yung pinapagawa ko, so pag may sinabi sila about sa requirement, most likely totoo yun. Maniwala kayo sa kanila, kundi magtatampo sila. :D**