Group Name: love124	**Section: ST1L**
Member 1: Myko Jefferson Javier	**Member 3: Franz Saragena**
Member 2: Justin Dayne Bryant Pena	**Member 4:**

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>	::=	[<comment>] [<function_def>] HAI <linebreak> <variable_section> <statement_list> <linebreak> KTHXBYE [<comment>] [<function_def>]
<var_section>	::=	WAZZUP <linebreak> <var_dec_list> BUHBYE <linebreak> | ε
<var_dec_list>		<declaration> <var_dec_list> | ε
<literal>	::=	numbr | numbar | yarn | troof | noob
<linebreak>	::=	\n
<statement_list>	::=	<statement> <line_break> <statement_list> | ε
<statement>	::=	<expression> | <conditional> |<loop> | <function_call> | <function_def> | <declaration> | <input> | <output>
<declaration>	::=	I HAS A <varident> <initialization>
<initialization>	::=	ITZ <expression> | ε
<comment>	::=	<single_line_comment> | <multiline_comment>
<single_line_comment>	::=	BTW text
<multiline_comment>	::=	OBTW <linebreak> text <linebreak> TLDR
<expression>	::=	<nestable_expr> | <non_nestable_expr>
<nestable_expr>	::=	<arithmetic_expr> | <boolean_nest> | <comparison>| <function_call> | <typecasting> | <literal> | <relational> | varident
<non_nestable_expr>	::=	<concatenation> | <boolean_non_nest>

<boolean_nest>	::=	BOTH OF <nestable_expr> AN <nestable_expr> | EITHER OF <nestable_expr> AN <nestable_expr> | WON OF <nestable_expr> AN <nestable_expr> | NOT <nestable_expr>
<boolean_non_nest>	::=	ALL OF <nestable_expr> AN <multi_expression_nestable> MKAY | ANY OF <nestable_expr> AN <multi_expression_nestable> MKAY
<multi_expression_nestable>	::=	AN <nestable_expr> <multi_expression_nestable> | ε
<arithmetic_expr>	::=	SUM OF <arithmetic_op> AN <arithmetic_op> | DIFF OF <arithmetic_op> AN <arithmetic_op> | PRODUKT OF <arithmetic_op> AN <arithmetic_op> | QUOSHUNT OF <arithmetic_op> AN <arithmetic_op> | MOD OF <arithmetic_op> AN <arithmetic_>
<arithmetic_op>	::=	<arithmetic_expr> | <literal> | varident
<comparison>	::=	BOTH SAEM <nestable_expr> AN <nestable_expr> | DIFFRINT <nestable_expr> AN <nestable_expr>
<relational>	::=	BOTH SAEM <nestable_expr> AN BIGGR OF <nestable_expr> AN <nestable_expr> | BOTH SAEM <nestable_expr> AN BIGGR OF <nestable_expr> AN <nestable_expr>| DIFFRINT <nestable_expr> AN SMALLR <nestable_expr> AN <nestable_expr>| DIFFRINT <nestable_expr> AN BIGGR OF <nestable_expr> AN <nestable_expr>
<typecasting>	::=	MAEK <nestable_expr> A <type_literal> | varident IS NOW A <type_literal>
<type_literal>	::=	NOOB | TROOF | NUMBAR | NUMBR | YARN
<function_call>	::=	I IZ funcident <param_list> MKAY
<param_list>	::=	YR varident <multi_param_list> | ε
<multi_param_list>	::=	AN YR varident <multi_param_list> | ε
<function_def>	::=	HOW IZ I funcident <param_list> <linebreak> <statement_list> <function_return> IF U SAY SO
<function_return>	::=	FOUND YR <expression> | GTFO | ε
<concatenation>	::=	SMOOSH <nestable_expr> <multi_expression_nestable>
<conditional>	::=	<if_case> | <switch_case>
<if_case>	::=	<nestable_expr>, O RLY? <linebreak> <if_true> <if_false> OIC
<if_true>	::=	YA RLY <linebreak> <statement_list> <linebreak>

<if_false>	::=	MEBBE <expression> <linebreak> <statement_list> <linebreak> <if_false> | NO WAI <linebreak> <statement_list> <linebreak> | ε
<switch_case>	::=	WTF? <linebreak> <case_block> <linebreak> OIC
<switch_exit>	::=	GTFO | ε
<case_block>	::=	OMG <literal> <linebreak> <statement_list> <linebreak> <switch_exit> <linebreak> <case_block> | OMGWTF <statement_list> <switch_exit> <linebreak>
<loop>	::=	IM IN YR loopident <loop_op> <loop_cond> <linebreak> <statement_list> <loop_exit> <optional_loop> <linebreak> IM OUTTA YR loopident
<optional_loop>	::=	<loop> | ε
<loop_op>	::=	UPPIN YR varident | NERFIN YR varident
<loop_cond>	::=	TIL <expression> | WILE <expression> | ε
<loop_exit>	::=	GTFO | ε
<input>	::=	GIMMEH varident
<output>	::=	VISIBLE <print_args>
<print_args>	::=	<expression> | <expression> + <print_args>

BONUS:
Array Implementation:

<type_literal>	::=	<type_literal> | UHS
<declaration>	::=	<declaration> | <array_declaration>
<array_declaration>	::=	I HAS A uhsident ITZ A UHS OF size
<array_operation>	::=	CONFINE varident IN uhsident AT index | DISCHARGE varident IN uhsident AT index | CONFINE <literal> IN uhsident AT index | DISCHARGE <literal> IN uhsident AT index | HIKE uhsident | SLIDE uhsident

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

LEXEME	Regular Expression
text	*
varident	^[a-zA-Z][a-zA-Z0-9_]*$
loopident	^[a-zA-Z][a-zA-Z0-9_]*$
funcident	^[a-zA-Z][a-zA-Z0-9_]*$
uhsicident	^[a-zA-Z][a-zA-Z0-9_]*$

size	^[0-9]+$
index	^[0-9]+$