Group Name: Hello, Hai, Kthxbye	**Section: S4L**
Member 1: Psymon Sez E. Arcedera	**Member 3: James Carl V. Villarosa**
Member 2: Aaron Jacob C. Perez	

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 <statement> linebreak KTHXBYE
<statement>	::=	<statement> linebreak <statement> | <statement> <comment> | <input> | <print> | <variable> | <expression> | <assignment> | <typecast> | <flowcontrol> | <function_definition> | <function_call> | GTFO
<comment>	::=	BTW <character> linebreak | linebreak OBTW <value_literal> TLDR
<input>	::=	GIMMEH varident
<print>	::=	VISIBLE varident linebreak | VISIBLE <expr> linebreak | VISIBLE <value_literal> linebreak
<variable>	::=	WAZZUP linebreak <declaration> linebreak BUHBYE
<declaration>	::=	I HAS A varident | I HAS A varident ITZ varident | I HAS A varident ITZ <expression>
<expression>	::=	<arithmetic>|<comparison>|<boolean>
<assignment>	::=	varident R <type_literal> | varident R varident | varident R <expression> | varident R MAEK varident <type_literal>
<arithmetic>	::=	SUM OF <operand> AN <operand> | DIFF OF <operand> AN <operand> | PRODUCKT OF <operand> AN <operand> | QUOSHUNT OF <operand> AN <operand> | MOD OF <operand> AN <operand> | BIGGR OF <operand> AN <operand> | SMALLR OF <operand> AN <operand>
<concatenation>	::=	SMOOSH <multioperand>

<comparison>	::=	BOTH SAEM <operand> AN <operand> | DIFFRINT <operand> AN <operand> | BOTH SAEM <operand> AN BIGGR OF <operand> <operand> | BOTH SAEM <operand> AN SMALLR OF <operand> <operand> | DIFFRINT <operand> AN BIGGR OF <operand> <operand> | DIFFRINT <operand> AN SMALLR OF <operand> <operand>
<boolean>	::=	BOTH OF <operand> AN <operand> | EITHER OF <operand> AN <operand> | WON OF <operand> AN <operand> | NOT <operand> | ALL OF <multioperand> MKAY | ANY OF <multioperand> MKAY
<operand>	::=	varident | <value_literal> | <expression>
<multioperand>	::=	<operand> | <operand> AN <multioperand>
<typecast>	::=	MAEK varident A <type_literal> | MAEK varident <type_literal> | varident IS NOW A <type_literal>
<flowcontrol>	::=	<if_then> | <switch_case> | <loop>
<if_then>	::=	O RLY? linebreak YA RLY linebreak <statement_return> linebreak OIC | O RLY? linebreak YA RLY linebreak <statement_return> linebreak NO WAI linebreak <statement_return> OIC
<switch_case>	::=	WTF? linebreak <switch> OMGWTF linebreak <statement_return> linebreak OIC
<switch>	::=	OMG <value_literal> linebreak <statement_return> linebreak |OMG <value_literal> linebreak <statement_return> linebreak <switch> | OMG <value_literal> linebreak <statement_return> linebreak GTFO
<loop>	::=	IM IN YR varident <loop_type> linebreak <statement_return> linebreak IM OUTTA YR varident
<loop_type>	::=	UPPIN YR varident TIL <comparison> | UPPIN YR varident WILE <comparison> | NERFIN YR varident TIL <comparison> | NERFIN YR varident WILE <comparison>
<function_definition>	::=	HOW IZ I varident <parameter> linebreak <statement_return> linebreak IF U SAY SO | HOW IZ I varident linebreak <statement_return> linebreak IF U SAY SO
<statement_return>		<statement> | <statement> <statement_return> | FOUND YR <expression> | FOUND <value_literal>
<parameter>	::=	YR varident | YR varident AN <parameter>]

<function_call>	::=	I IZ varident <call_parameter> MKAY
<call_parameter>	::=	YR <expression> | YR <expression> AN <parameter>]
<type_literal>	::=	numbr | numbar | yarn | troof
<value_literal>	::=	NUMBR Literal | NUMBAR Literal | YARN Literal | TROOF Literal

NEWLY-ADDED LEXEMES

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

LEXEME	Regular Expression
linebreak	\n