Group Name:	**Section:**
Member 1:	**Member 3:**
Member 2:	**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>	::=	HAI <linebreak> <statement> <linebreak> KTHXBYE
<statement>	::=	WAZZUP <linebreak> <declaration> BUHBYE <linebreak> <statement> | <expr> | <declaration>
<declaration>	::=	I HAS A varident | I HAS A varident ITZ <value>
<literal>	::=	numbr | numbar | yarn | troof | noob
<typecast>	::=	MAEK varident A <datatype> | varident IS NOW A <datatype>
<datatype>	::=	NUMBR | NUMBAR | YARN | TROOF | NOOB
<expr>	::=	<print> <linebreak> <expr> | <arithmetic> <linebreak> <expr> | <bool> <linebreak> <expr> | <assignment> <linebreak> <expr> | <comparison> <linebreak> <expr> | <conditional> <linebreak> <expr> | <loop> <linebreak> <expr> | <function> <linebreak> <expr> | <input> <linebreak> <expr> | <comment> <linebreak> <expr> | <concat> <linebreak> <expr> | <break> <linebreak> <expr> | <return> | ε
<conditional>	::=	WTF? <linebreak> <conditional> | OMG <value> <linebreak> <expr> <linebreak> <conditional>| OMGWTF <value> <linebreak> <expr> <linebreak> <conditional> | O RLY? <linebreak> <conditional> | YA RLY <linebreak> <expr> <linebreak> <conditional> |

		NO WAI <linebreak> <expr> <linebreak> <conditional> | OIC
<loop>	::=	IM IN YR loopident <loopop> YR varident TIL <comparison> <linebreak> <expr> <linebreak> IM OUTTA YR loopident | IM IN YR loopident <loopop> YR varident WILE <comparison> <linebreak> <expr> <linebreak> IM OUTTA YR loopident
<break>	::=	GTFO
<loopop>	::=	UPPIN | NERFIN
<function>	::=	(func declaration) HOW IZ I funcident <linebreak> <expr> linebreak> IF U SAY SO | HOW IZ I funcident <funcparam> <linebreak> <expr> <linebreak> IF U SAY SO | (func call) HOW IZ I funcident | HOW IZ I funcident <funcparam>
<funcparam>	::=	YR varident AN <funcparam> | YR varident
<return>	::=	FOUND YR <value> | GTFO
<comment>	::=	BTW commentstr | OBTW commentstr <linebreak> TLDR
<concat>	::=	SMOOSH <valconnect>
<input>	::=	GIMMEH varident
<print>	::=	VISIBLE varident | VISIBLE <expr> | VISIBLE <literal>
<arithmetic>	::=	SUM OF <value> AN <value> | DIFF OF <value> AN <value> | PRODUKT OF <value> AN <value> | QUOSHUNT OF <value> AN <value> | MOD OF <value> AN <value> | BIGGR OF <value> AN <value> | SMALLR OF <value> AN <value>
<comparison>		BOTH SAEM <value> AN <value> | DIFFRINT <value> AN <value>
<bool>	::=	BOTH OF <value> AN <value> | EITHER OF <value> AN <value> | WON OF <value> AN <value> | NOT <value> | ALL OF <valconnect> MKAY | ANY OF <valconnect> MKAY
<valconnect>	::=	<value> AN <valconnect> | <value>
<assignment>	::=	varident R <value>

<value>	::=	<arithmetic> | <bool> | <literal> | varident | funcident

NEWLY-ADDED LEXEMES

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

LEXEME	Regular Expression