`.....`

Group Name: Group EZR	**Section: B-4L**
Member 1: Johanna Eikou Teknomo	**Member 3: Senen Zyril D. Tamargo**
Member 2: Rio John C. Ducusin, Junior	**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 | <single comment> <linebreak> <program> <linebreak> <single comment> | <single comment> <linebreak> <program> | <program> <linebreak> <single comment> | hai <single comment> <linebreak> <statement> <linebreak> kthxbye <single comment> | hai <single comment> <linebreak> <statement> <linebreak> kthxbye | hai <linebreak> <statement> <linebreak> kthxbye <single comment> | <multi comment> <program> | <program> <multi comment>
<linebreak>	::=	\n
<single comment>	::=	btw comment <linebreak> <statement> | <statement> btw comment | <statement> <linebreak> btw comment | btw comment <linebreak> <statement> btw comment | btw comment <linebreak> <statement> <linebreak> btw comment | <statement> btw comment <linebreak> btw comment | btw comment <linebreak> <statement> btw comment <linebreak> btw comment
<multi comment>	::=	obtw <infinite comment> <linebreak> tldr | obtw <linebreak> <infinite comment> <linebreak> tldr | <multi comment> <single statement> | <multi comment> <linebreak> <statement> | <statement> <linebreak> <multi comment>
<infinite comment>	::=	comment <linebreak> <infinite comment> | comment <linebreak>
<statement>	::=	<if else> | <loop> | <assignment> | <switch case> | <typecast> |

		<recast> | <variable declaration> | <print> | <input>
<multi statement>	::=	<if else> | <loop> | <switch>
<single statement>	::=	<assignment> | <typecast> | <recast> | <variable declaration> | <print> | <input>
<if else>	::=	<expression> <linebreak> o rly? <linebreak> ya rly <linebreak> <statement> <linebreak> no wai <linebreak> <statement> <linebreak> oic | <expression> <linebreak> o rly? <linebreak> ya rly <linebreak> <statement> <linebreak> oic | <logic> <linebreak> o rly? <linebreak> ya rly <linebreak> <statement> <linebreak> no wai <linebreak> <statement> <linebreak> oic | <logic> <linebreak> o rly? <linebreak> ya rly <linebreak> <statement> <linebreak> oic
<expression>	::=	<arithmetic operation> | <concatenation> | <boolean operation> | <infinite operation>
<arithmetic operation>	::=	<add> | <subtract> | <multiply> | <divide> | <modulo> | <max> | <min>
<concatenation>	::=	smoosh <concatenation operand>
<concatenation operand>	::=	<operations operand> an <concatenation operand> | <operations operand>
<boolean operation>	::=	<not> | <and> | <or> | <xor>
<infinite operation>	::=	<infinite and> | <infinite or>
<literal>	::=	numbr | numbar | yarn | troof
<operations operand>	::=	<literal> | variableidentifier
<add>	::=	sum of <operations operand> an <operations operand>
<subtract>	::=	diff of <operations operand> an <operations operand>
<multiply>	::=	produkt of <operations operand> an <operations operand>
<divide>	::=	quoshunt of <operations operand> an <operations

		operand>
<modulo>	::=	mod of <operations operand> an <operations operand>
<max>	::=	biggr of <operations operand> an <operations operand>
<min>	::=	smallr of <operations operand> an <operations operand>
<not>	::=	not <operations operand>
<and>	::=	both of <operations operand> an <operations operand>
<or>	::=	either of <operations operand> an <operations operand>
<xor>	::=	won of <operations operand> an <operations operand>
<infinite and>	::=	all of <infinite operand> mkay
<infinite or>	::=	any of <infinite operand> mkay
<infinite operand>	::=	<infinite operand> an <infinite operand> | <operations operand> | <boolean operation>
<loop>	::=	im in yr loopidentifier <loop operation> yr variableidentifier wile <logic> <linebreak> <statement> <linebreak> im outta yr <loopidentifier> | im in yr loopidentifier <loop operation> yr variableidentifier til <logic> <linebreak> <statement> <linebreak> im outta yr <loopidentifier> | im in yr loopidentifier <loop operation> yr variableidentifier til <logic> <linebreak> <statement> <linebreak> gtfo <linebreak> im outta yr <loop identifier> | im in yr loopidentifier <loop operation> yr variableidentifier wile <logic> <linebreak> <statement> <linebreak> gtfo <linebreak> im outta yr <loop identifier>
<loop operation>	::=	uppin | nerfin
<logic>	::=	<equal> | <not equal>
<equal>	::=	both saem <logic operand> an <logic operand>
<not equal>	::=	diffrint <logic operand> an <logic operand>
<logic operand>	::=	<operations operand> | <expression>
<assignment>	::=	variableidentifier r <expression> | variableidentifier r <literal> | variableidentifier r variableidentifier
<switch case>	::=	wtf? <linebreak> <case> <linebreak> oic | wtf? <linebreak> <case> <linebreak> <default case> <linebreak> oic
<case>		omg <literal> <linebreak> <statement> | <case> <linebreak> <case>

<default case>	::=	omgwtf <linebreak> <statement>
<typecast>	::=	maek variableidentifier type | maek variableidentifier a type | maek <expression> type | maek <expression> a type
<recast>	::=	variableidentifier is now a type | variableidentifier r maek variableidentifier type | variableidentifier r maek <expression> type
<variable declaration>	::=	i has a variableidentifier | i has a variableidentifier itz <literal> | i has a variableidentifier itz variableidentifier | i has a variableidentifier itz <expression>
<print>	::=	visible <print operand>
<print operand>	::=	variableidentifier | <expression> | <literal> | <print operand> an <print operand>
<input>	::=	gimmeh variableidentifier

NEWLY-ADDED LEXEMES

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

LEXEME	Regular Expression
comment	^(.*)$
GTFO	^(GTFO)$
AN	^(AN)$
MKAY	^(MKAY)$