CMSC 129 Principles of Compiler Design **Second Semester AY 17-18**
Regular Expressions for LOLCODE Constructs **Project Requirement 01**

Name: Ciara Mae Gotis

IDENTIFIERS

PL Element	Regular Expression
Variable Identifier	^[\w]{1}[\w_\d]*$
Function Identifier	^[\w]{1}[\w_\d]*$
Loop Identifier	^[\w]{1}[\w_\d]*$

LITERALS

PL Element	Regular Expression
NUMBR Literal	^-?[\d]+$
NUMBAR Literal	^-?[\d]+.[\d]+$
YARN Literal	^“.*”$
TROOF Literal	^WIN$\|^FAIL$
TYPE Literal	((^NUMBR$)\|(^NUMBAR$)|(^YARN$)\|(^TROOF))$

KEYWORDS

PL Element	Regular Expression
HAI	^HAI$
KTHXBYE	^KTHXBYE$
BTW	^BTW$
OBTW	^BTW$
TLDR	^TLDR$
I HAS A	^I\s+ HAS\s+ A$
ITZ	^ITZ$
R	^R$
SUM OF	^SUM\s+OF$
DIFF OF	^DIFF\s+OF$
PRODUKT OF	^PRODUKT\s+OF$
QUOSHUNT OF	^QUOSHUNT$
MOD OF	^MOD\s+OF$
BIGGR OF	^BIGGR \s+OF$

SMALLR OF	^SMALLR\s+OF$
BOTH OF	^BOTH\s+OF$
EITHER OF	^EITHER\s+OF$
WON OF	^WON\s+OF$
NOT	^NOT$
ANY OF	^ANY\s+OF$
ALL OF	^ALL\s+OF$
BOTH SAEM	^BOTH\s+SAEM$
DIFFRINT	^DIFFRINT$
SMOOSH	^SMOOSH$
MAEK	^MAEK$
A	^A$
IS NOW A	^IS\s+NOW\s+A$
VISIBLE	^VISIBLE$
GIMMEH	^GIMMEH$
O RLY?	^O\s+RLY\?$
YA RLY	^YA\s+RLY$
MEBBE	^MEBBE$
NO WAI	^NO\s+WAI$
OIC	^OIC$
WTF?	^WTF\?$
OMG	^OMG$
OMGWTF	^OMGWTF$
IM IN YR	^IM\s+IN\s+YR$
UPPIN	^UPPIN$
NERFIN	^NERFIN$
YR	^YR$
TIL	^TIL$
WILE	^WILE$
IM OUTTA YR	^IM\s+OUTTA\s+YR$

LOLCODE GRAMMAR

Phrases enclosed by angle brackets (<,>) are abstractions. Words in small letters describe the lexemes that are already described by a regular expression (e.g. `varident` for variable identifiers, `yarn` for string literals, `troof` for boolean values, etc).

LHS	::=	RHS
<program>	::=	hai <numbar> <line break> <statement> <line break> kthxbye| hai <line break> <statement> <line break> kthxbye
<statement>	::=	**<comment>|** **<importation>|** <expression>| <output>| **<input>|** <variable_declaration>| <assignment>| <typecast>| <conditional>| <loop>| <statement> <line break> <statement>
<literal>	::=	numbr | numbar | yarn | troof | type | noob
<comment>	::=	btw <any character>| obtw <line break> <any character> <line break> TLDR|
<importation>	::=	CAN HAS <library> ?
<library>		STDIO | STRING | SOCKS | STDLIB
<expression>	::=	<comparison>| <arithmetic_operation>| varident| <concatenation>| <boolean operation>| <function call>| <literal>
<output>		visible <expression>
<input>	::=	gimmeh varident
<variable_declaration>	::=	i has a <variable identifier>| i has a <variable identifier> ITZ <variable initialization>|
<variable initialization>	::=	<expression>| A type
<assignment>	::=	varident R <expression>

`<comparison>`	::=	`both saem <expression> AN <expression>|`
		`diffrint <expression> AN <exprcompession>|`
		`biggr of <expression> AN <expression>|`
		`smallr of <expression> AN <expression>|`
		`both saem <expression> <expression>|`
		`diffrint <expression> <expression>|`
		`biggr of <expression> <expression>|`
		`smallr of <expression> <expression>|`

`<arithmetic_operation>`	::=	`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>|`

`<concatenation>`	::=	`smoosh <argument> MKAY|`
		`smoosh <argument>`

`<argument>`	::=	`<expression>|`
		`<expression> AN <argument>`

`<typecast>`	::=	`maek <expression> A type|`
		`maek <expression> type|`
		`<expression> IS NOW A type`

`<conditional>`	::=	`<expression> <line break> o rly? <line break> ya rly <line break>`
		`<statement> no wai <line break> <statement> oic|`
		`<expression> <line break> o rly? <line break> ya rly <line break>`
		`<statement> <line break>`
		`<else if> <line break>`
		`no wai <line break> <statement> oic|`
		`<expression><line break>wtf?<line break><case> <line break> oic`

`<else if>`	::=	`mebbe <line break> <statement>|`
		`mebbe <line break> <statement> <line break> <else if>`

`<case>`	::=	`wtf? <line break> omg <literal> <line break> <statement>`
		`<line_break> <case>|`
		`wtf? <line break> omg <literal> <line break> <statement>`
		`<line_break> gtfo <line_break> <case>|`
		`wtf? <line break> omg <literal> <line break> <statement>|`
		`wtf? <line break> omg <literal><line break> <statement>`
		`<line_break> gtfo|`
		`wtf? <line break> omgwtf <literal> <line break> <statement>`

`<loop>`	::=	`Lo op_identifier <loop operation> yr varident <loop rule>`
		`<expression> <line break> <statement> <line break>`

		loop_identifier
<loop operation>	::=	uppin| nerfin|
<loop rule>	::=	til| wile
<function definition>	::=	how iz i function_identifier <line break> <statement> if you say so| how iz i function_identifier <function parameter> <line break> <statement> if you say so| how iz i function_identifier <line break> <statement> <function termination>if you say so| how iz i function_identifier <function parameter> <line break> <statement> <function termination> if you say so
<function parameter>		yr <expression>| yr <expression> AN <function parameter>| yr <expression> <function parameter>
<function termination>	::=	FOUND yr <expression>| GTFO
<boolean operation>	::=	BOTH OF <expression> <expression>| BOTH OF <expression> AN <expression>| EITHER OF <expression> <expression> | EITHER OF <expression> AN <expression> | WON OF <expression> <expression> | WON OF <expression> AN <expression>| NOT <expression>
<function call>		I IZ function_identifier mkay| I IZ function_identfier <function parameter> mkay