CMSC 124
Design and Implementation of Programming Languages

Project Requirement 01
Regex

Group Name:	Group 2	Section:	T – 1L
Member 1:	Rivera, Marianne Louise	Member 3:	Serato, Mike Edward
Member 2:	Edora, Nixon Jr.		

Identifiers

PL Element	Regular Expression
Variable Identifier	^[a-zA-Z]+$
Function Identifier	^\s*(HOW IZ I){1}\s+[a-zA-Z]+$
Loop Identifier	^\s*(IM IN YR){1}\s+[a-zA-Z]+.*(IM OUTTA YR)\s+[a-zA-Z]+$

Literals

PL Element	Regular Expression
NUMBR Literal	^\d+$
NUMBAR Literal	^\d*\.\d+$
YARN Literal	^".+"$
TROOF Literal	^(WIN|FAIL)$
TYPE Literal	^(NUMBR|NUMBAR|YARN|TROOF|BUKKIT)$

Keywords

PL Element	When/how to use?	Regular Expression
HAI	Beginning of every code	^(HAI)\s+$
KTHXBYE	End of every code	^\s*(KTHXBYE)\s*$
BTW	Used for declaring comments (single line)	^\s*(BTW)\s+.*$
OBTW	Used for declaring multi-lined comments	^\s*(OBTW)\s+.*$
TLDR	Used for declaring multi-lined comments	^\s*(TLDR)\s+.*$
I HAS A	Used for declaring variables	^\s*(I HAS A)\s+[a-zA-Z]+\s+((ITZ)\s+(\d+|\d*\.\d+|".+"|(WIN|FAIL))|(ITZ A)\s+(NUMBR|NUMBAR|YARN|TROOF|BUKKIT)|(ITZ LIKE A)\s+[a-zA-Z]+)?$
ITZ	Used for assigning a value to a variable directly after declaring it	^\s*(I HAS A)\s+[a-zA-Z]+\s+((ITZ)\s+(\d+|\d*\.\d+|".+"|(WIN|FAIL))|(ITZ A)\s+(NUMBR|NUMBAR|YARN|TROOF|BUKKIT)|(ITZ LIKE A)\s+[a-zA-Z]+)$
R	Used for assigning a value to a variable	^\s*[a-zA-Z]+\s+(R)\s+(\d+|\d*\.\d+|".+"|(WIN|FAIL))$

SUM OF	Calculates the sum of two expressions	^\s*(SUM OF)([a-zA-Z]+| \d+| \d*\.\d+|”.+”|(WIN|FAIL))(AN) ([a-zA-Z]+| \d+| \d*\.\d+|”.+”|(WIN|FAIL))$
DIFF OF	Calculates the difference of two expressions	^\s*(DIFF OF)([a-zA-Z]+| \d+| \d*\.\d+|”.+”|(WIN|FAIL))(AN) ([a-zA-Z]+| \d+| \d*\.\d+|”.+”|(WIN|FAIL))$
PRODUKT OF	Multiplication	^(PRODUKT OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
QUOSHUNT OF	Division	^(QUOSHUNT OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
MOD OF	Division remainder (modulo)	^(MOD OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
BIGGR OF	Maximum of 2 numbers	^(BIGGR OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
SMALLR OF	Minimum of 2 numbers	^(SMALLR OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
BOTH OF	Logical “and”	^(BOTH OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
EITHER OF	Logical “or”	^(EITHER OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
WON OF	Logical “xor”	^(WON OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
NOT	Logical “not”	^(NOT)\s+[A-Za-z0-9]+$
ALL OF	Logical “and” for arbitrary no. of arguments	^(ALL OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
ANY OF	Logical “or” for arbitrary no. of arguments	^(ANY OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
BOTH SAEM	Equality	^(BOTH SAEM)\s+[A-Za-z0-9]+\s+(AN)\s+(BIGGR|SMALLR)\s+(OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
DIFFRINT	Inequality	^(DIFFRINT)\s+[A-Za-z0-9]+\s+(AN)\s+(BIGGR|SMALLR)\s+(OF)\s+[A-Za-z0-9]+\s+(AN)\s+[A-Za-z0-9]+$
SMOOSH	String concatenation	^(SMOOSH)\s+.+$
MAEK	Converts value to a data type	^(MAEK)\s+[A-Za-z0-9]+\s+(A)\s+[A-Za-z0-9]+$
A	For converting a value of the given expression to the given data type	^[A-Za-z0-9]+\s+(A)\s+[A-Za-z0-9]+$
IS NOW A	For converting the type of the variable	^[A-Za-z0-9]+\s+(IS NOW A)\s+[A-Za-z0-9]+$
VISIBLE	For printing	^(VISIBLE)\s+.+$

GIMMEH	Reads an input string	^(GIMMEH)\s+[A-Za-z]+$
O RLY?	An if statement	/O\sRLY?/
YA RLY	Executed if WIN(true) is evaluated	/YA\sRLY/
MEBBE	If expression following MEBBE is WIN, performs block	/MEBBE/
NO WAI	Executed if false is evaluated	/NO\sWAI/
OIC	END-IF	/OIC/
WTF?	LOLCODE equivalent of switch construct	/WTF?/
OMG	Comparison block`	/OMG/
OMGWTF	The default case for WTF?	/OMGWTF/
IM IN YR	While statement	/IM\sIN\sYR/
UPPIN	increasing(used in IM IN YR)	/UPPIN/
NERFIN	decreasing(used in IM IN YR)	/NERFIN/
YR	The value of the expression	/YR/
TIL	Evaluates the expression as TROOF	/TIL/
WILE	Converse of TIL	/WILE/
IM OUTTA YR	Exits the loop	/IM\sOUTTA\sYR/