**Mini-Language Specification**

Alphabet

* A-Z, a-z (Uppercase and lowercase letters)
* 0-9 (Digits)
* \_ (Underscore)

Operators

* +, -, \*, /, % ( Addition, substraction, multiplication, division, …- arithmetic)
* ===, =!=, >>, <<, >=, <= (Equality, inequality, greater, lower, greater or equal, lower or equal – relational)
* &&&, |||, ! (And, or, not – logical)
* = (Assignment)

Separators

* ;
* ,
* “ “
* {
* }
* (
* )

Keywords

* Read
* Write
* If
* Otherwise
* For
* While
* Break
* Integer
* String
* Character
* Array
* Return

Identifiers

* Identifier = letter {letter | digit} | letter
* Letter = A | B | … | Z | a | b| … | z
* Digit = 0 | 1 | 2 | … | 9

Constants

* Integer = 0 | 1 | 2 | … | 9 {digit}
* Character = ‘letter’ | ‘digit’
* String = “{letter | digit}”

Tokens:

* Identifier
* Constant
* [
* ]
* {
* }
* (
* )
* ;
* :
* ,
* <<
* >>
* ===
* =!=
* !
* &&&
* |||
* =
* +
* -
* \*
* /
* %
* Character
* Integer
* String
* Array
* If
* Otherwise
* For
* While
* Break
* Return
* Read
* Write
* Start

Syntax

* Declaration = type “ “ identifier
* Simple\_type = “integer” | “string” | “character”
* Array\_declaraction = simple\_type “ “ “array” “[“ integer “]”
* Type = simple\_type | array\_declaration
* Compound\_statement = “{“ statement\_list “}”
* Statement\_list = statement | statement “;” statement\_list
* Statement = simple\_statement | struct\_statement
* Simple\_statement = assign\_statement | io\_statement | declaration
* Struct\_statement = compound\_statement | if\_statement | while\_statement | for\_statement
* If\_statement = “if” condition statement [“otherwise” statement]
* For\_statement = “for” “(“ “number” assign\_statement “;” condition “;” assign\_statement “)” statement
* While\_statement = “while” condition statement
* Assign\_statement = identifier “=” expression
* Expression = [expression (“+”|”-“)]term
* Term = term (“\*”|”/”) factor | factor
* Factor = “(“ expression “)” | integer | identifier | identifier| “[“ integer “]”
* Io\_statement = (“read” IDENTIFIER) | (“write” (identifier | constant))
* Condition = “(“ expression relation expression “)”
* Relation = “<<” | “<=” | “===” | “=!=” | “>=” | “>>”