

Language Grammar & Descriptions

Table 1 ... Token Conversions

IN RAW CODE	AS TOKEN
func	FUNC
<some bool or int>	VAL(<some bool or int>)
<some id>	ID("<some id>")
true	TRUE
false	FALSE
!	NOT
&&	AND
	OR
==	EQ
!=	NOT_EQ
<	LT
>	GT
+	ADD
-	SUB
*	MULT
/	DIV
=	ASSIGN
let	LET
if	IF
else	ELSE
while	WHILE
return	RETURN
write	WRITE
[BRACE_L
]	BRACE_R
(PAREN_L
)	PAREN_R
;	SEMICOLON
,	COMMA
	ERROR
	EOI
	BLOCK
	PARAMS
read	READ
write	WRITE
print	PRINT

SYNTAX

```

<func> -> FUNC ID PAREN_L <parameters> PAREN_R <block>
<parameters> -> (<parameter>)*
<parameter> -> ID || (ID COMMA)
<block> -> BRACE_L <statements> BRACE_R

<let> -> LET ID SEMICOLON

<if> -> IF ID <relational operator> <point of comparison>
<block>
<relational operator> -> EQ || LT || GT
<point of comparison> -> ID || LIT_INT(<some int>) ||
LIT_BOOL(<some bool>)
<block> -> BRACE_L <statements> BRACE_R

<else> -> ELSE <block>
<block> -> BRACE_L <statements> BRACE_R

<while> -> WHILE ID <relational operator> <point of
comparison> <block>
<relational operator> -> EQ || LT || GT
<point of comparison> -> ID || LIT_INT(<some int>) ||
LIT_BOOL(<some bool>)
<block> -> BRACE_L <statements> BRACE_R

<return> -> RETURN <return body> SEMICOLON
<return body> -> <some value> (<operator> <some value>)*
<operator> ADD || SUB || MULT || DIV
<some value> -> ID || LIT_INT(<some int>) || LIT_BOOL(<some
bool>)

<write> -> WRITE <function call> SEMICOLON
<function call> -> ID PAREN_L <parameters> PAREN_R
<parameters> -> (<parameter>)*
<parameter> -> ID || (ID COMMA)

<ID> -> ID ASSIGN <return body> SEMICOLON
<id body> -> <some value> (<operator> <some value>)*
<some value> -> ID || LIT_INT(<some int>) || LIT_BOOL(<some
bool>)
<operator> ADD || SUB || MULT || DIV

```