```
< program >     ::= BEGIN < block-alt >
                          | NUMBER ID < program-alt >
                          | CHARACTER ID < program-alt >
                          | BOOL ID < program-alt >
                          | STRING ID < program-alt >
                          | < fun-declist > < block >

< program-alt > ::=  < var-declist2 > < var-declist > < fun-declist > < block >

< fun-declist > ::=  DEF < fun > < fun-declist' >

< fun-declist' > ::= DEF < fun > < fun-declist’ >
                          | ""

< fun >         ::= PROC ID LPAREN < variable_choice_fun >
                  | NUMBER ID LPAREN < variable_choice_fun >
                  | CHARACTER ID LPAREN < variable_choice_fun >
                  | BOOL ID LPAREN < variable_choice_fun >
                  | STRING ID LPAREN < variable_choice_fun >

< variable_choice_fun > ::= < param-list > RPAREN < block >
                                | RPAREN < block >

< param-list >  ::= NUMBER < variable_choice_param1 > < variable_choice_param2 >
                  | CHARACTER < variable_choice_param1 > < variable_choice_param2 >
                  | BOOL < variable_choice_param1 > < variable_choice_param2 >
                  | STRING < variable_choice_param1 > < variable_choice_param2 >

< variable_choice_param1 > ::= ID
                        | CLOSED_BRACKET ID

< variable_choice_param2 > ::= COMMA < param-list >
                        | ""

< block >       ::= BEGIN < block-alt >

< block-alt >   ::= NUMBER ID < var-declist2 > < var-declist > < stmnt-list > END
                          | CHARACTER ID < var-declist2 > < var-declist > < stmnt-list > END
                          | BOOL ID < var-declist2> < var-declist > < stmnt-list > END
                          | STRING ID < var-declist2> < var-declist > < stmnt-list > END
                          | < stmnt-list > END
```

```
< var-declist > ::= NUMBER ID < var-declist2 > < var-declist >
                    | CHARACTER ID < var-declist2 > < var-declist >
                    | STRING ID < var-declist2 > < var-declist >
                    | BOOL ID < var-declist2 > < var-declist >
                    | ""

< var-declist2 > ::= LBRACKET < bounds > RBRACKET
                   | ""

< bounds >     ::=  INTLIT < bounds' >

< bounds' >    ::=  COMMA INTLIT < bounds' >
                  | ""

< stmnt-list >  ::= < stmnt > < stmnt-list-alt >

< stmnt-list-alt > ::= < stmnt > < stmnt-list-alt >
                    | ""

< stmnt >       ::= ID < stmnt-alt >
                |  IF  < condition > < block > < branch-alt >
                | WHILE < condition > < block >
                | LPAREN < Expression > RPAREN
                | INTLIT < Expression >
                | FLOATLIT < Expression >
                | CHARLIT < Expression >
                | STRINGLIT < Expression >
                | PRINT < arg-list >
                | READ < ref-list >
                | BREAK

                | < return >

< stmnt-alt >   ::= ASSIGN < stmnt-alt3 >
          | SWAP < ref >
          | LBRACKET < arg-list > RBRACKET < stmnt-alt2 >
          | < ref-alt >
          | LPAREN < call-alt >
          | CLOSED_BRACKET ASSIGN SPLIT LPAREN < split-alt > RPAREN STRINGLIT

< stmnt-alt2 >  ::= ASSIGN < expr >
                | SWAP < ref >

< stmnt-alt3 > ::= < expr >
```

```
                    | < import >

< branch-alt >  ::= ELSE < block >
                              | ""

< condition >   ::= < expr > < condition-alt >

< condition-alt > ::= EQUAL < expr >
                | NOT_EQUAL_TO < expr>
                | LESS THAN < expr >
                | LESS_THAN_OR_EQUAL < expr >
                | GREATER_THAN < expr >
                | GREATER_THAN_OR_EQUAL < expr >

< expr > ::= < term > < expr' >

< expr' > ::= PLUS < term > < expr' >
                | MINUS < term > < expr' >
                | ""

< term >        ::= < factor > < term' >

< term'>        ::= TIMES < factor > < term' >
                | DIVIDE < factor > < term' >
                | ""

< factor >      ::= < exponent > < factor' >

< factor' >     ::= POW < exponent > < factor' >
                | ""

< exponent >    ::= LPAREN < expr > RPAREN
                | ID < exponent-alt >
                | INTLIT
                | FLOATLIT
                | CHARLIT
                | STRINGLIT

< exponent-alt > ::= < ref-alt >
                | LPAREN < call-alt >

< arg-list >    ::= < expr > < arg-list-alt >

< arg-list-alt > ::= COMMA < arg-list >
```

| ""

< ref-list > ::= < ref > < ref-list-alt >

< ref-list-alt > ::= COMMA < ref-list >
| ""

< ref > ::= ID < ref-alt >

< ref-alt > ::= LBRACKET < arg-list > RBRACKET
| ""

< call-alt > ::= < arg-list > RPAREN
| RPAREN

< return > ::= RETURN < expression >

< import > ::= IMPORT < import - alt >

< import - alt > :: = STRINGLIT

| < ref >

< split-alt > ::= < ref >

| CHARLIT

| STRINGLIT

Lexer Grammar (Regular)

Token	Rule
SLASH_DOUBLE_QUOTE	\"
SINGLE_QUOTE	'
GREATER_THAN	>
GREATER_THAN_OR_EQUAL	>=
LESS_THAN	<
LESS_THAN_OR_EQUAL	<=
NOT_EQUAL_TO	~=
SWAP	:=:
ASSIGN	:=
LBRACKET	[
RBRACKET	]
CLOSED_BRACKET	[]

COMMA	,
EQUAL	=
PLUS	+
MINUS	-
TIMES	*
DIVIDE	/
POW	**
LPAREN	(
RPAREN	)
ID	([a-zA-Z]|_)[a-zA-Z0-9_]*
SINGLE_CHAR	Any character, with normal escapes \n, \t, \', \"
CHARACTERS	a string of SINGLE_CHAR
STRINGLIT	SLASH_DOUBLE_QUOTE CHARACTERS SLASH_DOUBLE_QUOTE
CHARLIT	SINGLE_QUOTE SINGLE_CHAR SINGLE_QUOTE
INTLIT	a string of digits
FLOATLIST	INTLIT . INTLIT
PROC	PROC
BEGIN	BEGIN
END	END
NUMBER	NUMBER
CHARACTER	CHARACTER
IF	IF
ELSE	ELSE
WHILE	WHILE
PRINT	PRINT
READ	READ
RETURN	RETURN
IMPORT	IMPORT
FOR	FOR
AS	AS
O	O
I	I
BREAK	BREAK
SPLIT	SPLIT
BOOL	BOOL
STRING	STRING
DEF	DEF