LEXER REGEX AND PARSER GRAMMATICAL RULES

```
LBLOCK              = '{'
 RBLOCK = '}'
 SEMICOLON. = ';
' LPAREN. =  '('
 RPAREN.             =   ')'
 PLUS.               =   '+'
 EQUAL               =   '='
COMPARE.            =   '=='
LESSEREQ.           = '<='
 NOTEQ.             = '>='
GREATEREQ.          = '!='
 IF                 = 'if'
 ELSE.              = 'else'
 WHILE.             = 'while'
 INT                ='int'
 VOID               ='void'
 NUM= '\d*'
LESSER. = '<'
GREATER. = '>'
MINUS.              = '-'
 ENDFILE.           =   '\$'
 COMMA              =   ','
 TIMES              =   '\*'
 DIVIDE             = '/'
  LBRACKET. = '\['
  RBRACKET. = '\]'
  COMMENT='\/\*(\*(?!\/)|[^*])*\*\/*'
    ID= '[a-zA-Z][a-zA-Z0-9]*'
RESERVED.           ='if|else|int|void|return'
```

```
X->Start
Start->Declarations
Declarations->fun_declara | var_declara
Declarations->EOF
Fun _declara | var_declara->type
Type->spec ID ; | spec ID {NUM}
Spec> void | int
;
Fun_declara->spec(parameter) multstatement
Parameter->parameter list
Parameter list->parameter parameter
Multstatement->statement_list
Statement_list->statemnt,stratemnt
Statemnt->expression|compound|selection|return
Expression->exp ; | ; | IF (exp) | while(exp) | var=exp 1exp operator factor
Return->return exp ;
Operator->GREATER,NE,EQ,COMPARE,LESS,PLUS,MINUS,TIMES,DIVIDE
Factor ->(exp) | ID | NUM | CALL
Call->ID(args)
Args->VOID, list
List->ag,arg
```

A01366101

Ma. Fernanda Delgado Radillo