Grammar for CS 152 – Phase 2 Project

Terminal -> UPPER CASE LETTERS or “symbols”

Non-terminal -> lower case letters

---

program -> function program | ε

declaration -> identifiers “:” INTEGER | identifiers “:” ARRAY “[“ NUMBER “]” OF INTEGER

identifiers -> ident “,” identifiers | ident

ident -> IDENTIFIER

function -> FUNCTION IDENTIFIER “;” BEGIN_PARAMS parameters

parameters -> declaration “;” parameters | END_PARAMS BEGIN_LOCALS locals

locals -> declaration “;” locals | END_LOCALS BEGIN_BODY bstatements

bstatements -> statement “;” bstatements | statement “;” END_BODY

statement1 -> var “: =” expression

statement2 -> IF bool_expr THEN statement_loop2

statement_loop2 -> statement “;” statement_loop2 | statement “;” ELSE statement_loop2b | statement “;” ENDIF

statement_loop2b -> statement “;” statement_loop2b | statement “;” ENDIF

statement3 -> WHILE bool_expr BEGINLOOP statement_loop3

statement_loop3 -> statement “;” statement_loop3 | statement “;” ENDLOOP

statement4 -> DO BEGINLOOP statement_loop4

statement_loop4 -> statement “;” statement_loop4 | statement “;” ENDLOOP WHILE bool_expr

statement5 -> READ var_loop

var_loop -> var “,” var_loop | var “,” | var

statement6 -> WRITE var_loop

statement7 -> CONTINUE

statement8 -> RETURN expression

statement -> statement1 | statement2 | statement3 | statement4 | statement5 | statement6 | statement7 | statement8

bool_expr -> relation_and_expr | relation_and_expr OR bool_expr_loop

bool_expr_loop -> relation_and_expr | relation_and_expr OR bool_expr_loop

relation_and_expr -> relation_expr | relation_expr AND relation_expr_loop

relation_expr_loop -> relation_expr | relation_expr AND relation_expr_loop

relation_expr -> NOT relation_exprS | relation_exprS

relation_exprS -> expression comp expression | TRUE | FALSE | “(“ bool_expr “)”

comp -> “==” | “<>” | “<” | “>” | “<=” | “>=”

expression -> mult_expr “+” expression | mult_expr “-“ expression | mult_expr

mult_expr -> term “*” mult_expr | term “/” mult_expr | term “%” mult_expr | term

term -> “-“ term1 | term1 | term2

term1 -> var | NUMBER | “(“ expression “)”

term2 -> IDENTIFIER “(“ expression_loop “)”

expression_loop -> expression | expression “,” expression_loop

var -> IDENTIFIER | IDENTIFIER “[“ expression “]”