1. Remove left factoring

program → decl_list

decl_list → decl_list decl | decl

decl → var_decl | fun_decl

var_decl → type_spec IDENT var_decl_

var_decl_ →; | [ ] ;

type_spec → VOID | BOOL | INT | FLOAT

fun_decl → type_spec IDENT (params) compound_stmt

params → param_list | VOID

param_list → param_list , param | param

param → type_spec IDENT param_

param_ → ε | []

stmt_list → stmt_list stmt | ε

stmt → expr_stmt | compound_stmt | if_stmt | while_stmt |

return_stmt | break_stmt

expr_stmt → expr ; | ;

while_stmt → WHILE (expr) stmt

compound_stmt → { local_decls stmt_list }

local_decls → local_decls local_decl | ε

local_decl → type_spec IDENT local_decl_

local_decl_ → ; | [ ] ;

if_stmt → IF (expr) stmt if_stmt_

if_stmt_ → ε | ELSE stmt

return_stmt → RETURN return_stmt_

return_stmt_ → ; | expr ;

expr → IDENT expr_IDENT

→ expr expr_expr

→ ! expr | - expr | + expr

→ (expr)

→ BOOL_LIT | INT_LIT | FLOAT_LIT | NEW type_spec [expr]

expr_IDENT → = expr | [expr] = expr | [expr] | (args) | . size | ε

expr_expr → OR expr | EQ expr | NE expr | LE expr | < expr | GE expr | > expr | AND expr | + expr | - expr | * expr | / expr | % expr

arg_list → arg_list , expr | expr

args → arg_list | ε

2. Remove left recursion

program → decl_list

decl_list → decl_list decl | decl (before removing left recursion)

decl_list → decl _decl_list (after removing left recursion)

_decl_list → decl _dec_list | ε

decl → var_decl | fun_decl

var_decl → type_spec IDENT var_decl_

var_decl_ →; | [ ] ;

type_spec → VOID | BOOL | INT | FLOAT

fun_decl → type_spec IDENT (params) compound_stmt

params → param_list | VOID

param_list → param_list , param | param

param_list → param _param_list

_param_list → , param _param_list | ε

param → type_spec IDENT param_

param_ → ε | []

stmt_list → stmt_list stmt | ε

stmt_list → _stmt_list

_stmt_list → stmt _stmt_list | ε

stmt → expr_stmt | compound_stmt | if_stmt | while_stmt | return_stmt | break_stmt

expr_stmt → expr ; | ;

while_stmt → WHILE (expr) stmt

compound_stmt → { local_decls stmt_list }

local_decls → local_decls local_decl | ε

local_decls → _local_decls

_local_decls → local_decl _local_decls | ε

local_decl → type_spec IDENT local_decl_

local_decl_ → ; | [ ] ;

if_stmt → IF (expr) stmt if_stmt_

if_stmt_ → ε | ELSE stmt

return_stmt → RETURN return_stmt_

return_stmt_ → ; | expr ;

expr → expr expr_expr | IDENT expr_IDENT | ! expr | - expr | + expr | (expr) | BOOL_LIT | INT_LIT | FLOAT_LIT | NEW type_spec [expr]

expr → IDENT expr_IDENT _expr | ! expr _expr | - expr _expr | + expr _expr | (expr) _expr | BOOL_LIT _expr | INT_LIT _expr | FLOAT_LIT _expr | NEW type_spec [expr] _expr

_expr → expr_expr _expr | ε

expr_IDENT → = expr | [expr] = expr | [expr] | (args) | . size | ε

expr_expr → OR expr | EQ expr | NE expr | LE expr | < expr | GE expr | > expr | AND expr | + expr | - expr | * expr | / expr | % expr

arg_list → arg_list , expr | expr

arg_list →expr _arg_list

_arg_list →, expr _arg_list| ε

args → arg_list | ε

3. Final grammar

//start -> program | stmt_list

program → decl_list

decl_list → decl _decl_list

_decl_list → decl _dec_list | ε

decl → type_spec IDENT decl_

decl_ -> var_decl_ | (params) compound_stmt

var_decl_ →; | [ ] ;

type_spec → VOID | BOOL | INT | FLOAT

params → param _param_list | VOID

_param_list → , param _param_list | ε

param → type_spec IDENT param_

param_ → ε | []

stmt_list → stmt stmt_list | ε

stmt → expr_stmt | compound_stmt | if_stmt | while_stmt | return_stmt | break_stmt

expr_stmt → expr ; | ;

while_stmt → WHILE (expr) stmt

compound_stmt → { local_decls stmt_list }

var_decl → type_spec IDENT var_decl_

local_decls → var_decl   local_decls | ε

if_stmt → IF (expr) stmt if_stmt_

if_stmt_ → ε | ELSE stmt

return_stmt → RETURN return_stmt_

return_stmt_ → ; | expr ;

expr → IDENT expr_IDENT _expr | ! expr _expr | - expr _expr | + expr _expr | (expr) _expr | BOOL_LIT _expr | INT_LIT _expr | FLOAT_LIT _expr | NEW type_spec [expr] _expr

_expr → expr_expr _expr | ε

expr_IDENT → = expr | [expr] = expr | [expr] | (args) | . size | ε

expr_expr → OR expr | EQ expr | NE expr | LE expr | < expr | GE expr | > expr | AND expr | + expr | - expr | * expr | / expr | % expr

arg_list →expr _arg_list

_arg_list →, expr _arg_list| ε

args → arg_list | ε