program → declaration-list

declaration-list → decl {decl}

decaration → (int | void) ID var’fun’

var’fun’ → (; | [NUM]) | ((params) compound-stmt)

params → param-list | void

param-list → param {,param}

param → int ID [[]]

compound-stmt → {local-declarations statement-list}

local-declarations → {var-declaration}

statement-list → {statement}

statement → expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt

expression-stmt → [expression];

selection-stmt → if (expression) statement [else statement]

iteration-stmt → while (expression) statement

return-stmt → return [expression];

expression → NUM simple-expression’ | (expression) simple-expression’ | ID expression’

expression’ → =expression | [expression] expression’’| (args) simple-expression’ | simple-expression’

expression’’→ expression | simple-expression’

var-call → [ expression] | (args)

simple-expression’ → additive-expression’ [relop additive-expression]

relop → <= | < | > | => | == | !=

additive-expression → {addop term}

additive-expression’ → term’ {addop term}

addop → +|-

term → factor {mulop factor}

term’ → {mulop factor}

mulop → \*|/

factor → (expression) | ID varCall | NUM

args → arg-list

arg-list → expression {, expression}