program -> declaration-list EOF

declaration-list -> declaration declaration-list | EPSILON

declaration -> type-specifier s1

s1 -> ID s10

s10 -> ; | [ NUM ] ; | ( params ) compound-stmt

type-specifier -> int | void

params -> int ID s3 s2 | void s11

s11 -> ID s3 s2| EPSILON

s2 -> , param s2 | EPSILON

param -> type-specifier ID s3

s3 -> [ ] | EPSILON

compound-stmt -> { declaration-list statement-list }

statement-list -> statement statement-list | EPSILON

statement -> expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt |switch-stmt

expression-stmt -> expression ; | continue ; | break ; | ;

selection-stmt -> if ( expression ) statement else statement

iteration-stmt -> while ( expression ) statement

return-stmt -> return s16

s16 -> ; | expression ;

switch-stmt -> switch ( expression ) { case-stmts default-stmt }

case-stmts -> case-stmt case-stmts | EPSILON

case-stmt -> case NUM : statement-list

default-stmt -> default : statement-list | EPSILON

expression -> ID s12 | ( expression ) s8 s7 s6 | | NUM s8 s7 s6 | - factor s8 s7 s6 | + factor s8 s7 s6

s12 -> = expression | [ expression ] s13 | ( args ) s8 s7 s6 | s8 s7 s6

s13 -> = expression | s8 s7 s6

var -> ID s5

s5 -> [ expression ] | EPSILON

simple-expression -> additive-expression s6

s6 -> relop signed-factor s8 s7 expression | EPSILON

relop -> < | ==

additive-expression -> term s7

s7 -> addop term s7 | EPSILON

addop -> + | -

term -> signed-factor s8

s8 -> \* signed-factor s8 | EPSILON

signed-factor -> factor | + factor | - factor

factor -> ( expression ) | ID s15 | NUM

s15 -> s5 | ( args )

call -> ID ( args )

args -> arg-list | EPSILON

arg-list -> expression s9

s9 -> , expression s9 | EPSILON