Program -> external\_declaration

external\_declaration-> Func\_definitions | external\_decl\_stmt

Func\_definitions -> {Func\_definition}+

Func\_definition -> type id “(“ param\_type\_list “)” block

external\_decl\_stmt -> <empty> | type declarators “;”

Type -> int | float | double | char | long | ………….

Param\_type\_list -> type id {type\_param}\* | <empty>

Type\_param -> “,” type id | <empty>

Block -> “{“ stmts “}”

Stmts -> decl\_stmt | if\_stmt | while\_stmt | for\_stmt | dowhile\_stmt | switch\_stmt | return\_stmt | assign | <empty>

Decl\_stmt -> <empty> | type declarators “;”

Declarators -> declarator { “,” declarator }

Declarator -> {Pointer}? id { arrayDeclarator} ? assignForDecl

Pointer -> “\*” {pointer}?

arrayDeclarator -> “[“ {const\_expr}? “]” {arrayDeclarator}?

const\_expr -> conditional\_expr

conditional\_expr -> logical\_expr | logical\_expr “?” expr : conditional\_expr

assignForDecl -> <empty> | “=” expr

factor -> num | “(“ expr “)” | abstract\_declarator

abstract\_declarator-> id | id “[“ expr “]” | id “(“ param\_list “)” | id “->” abstract\_declarator

| id “.” abstract\_declarator

param\_list -> <empty> | param { “,” param }\*

param -> expr

if\_stmt -> “if” “(“ expr “)” compound\_stmt

compound\_stmt -> block | stmt

while\_stmt -> “while” “(“ expr “)” compound\_stmt

assign -> Declarator “=” expr

expr -> assignment\_expr {“,” assignment\_expr }+

assignment\_expr -> conditional\_expr | unary\_expr assgnment\_op assignment\_expr

assgnment\_op -> “=” | “+=” | …

unary\_op -> “&” | “+” | “\*” | …

unary\_expr -> postfix\_expr | “++” unary\_expr | “—“ unary\_expr | unary\_op cast\_expr

cast\_expr -> unary\_expr | “(“ type\_name “)” cast\_expr

postfix\_expr -> primary\_expr { postfix\_op }+

postfix\_op -> “[“ expr “]” | “(“ assignment\_expr “)” | “->” id | “.” Id | “++” | “—“

primary\_expr -> id | constant | string | “(“ expr “)”

constant -> integer\_const | char\_const | floating\_const | enum\_const