The Context Free Grammar for C language can be given by G = (V, T, S, P):
where:

V = set of non-terminals
= {program_unit, translation_unit, external_decl, function_definition, decl, decl_list, decl_specs, storage_class_spec, type_spec, type_qualifier, struct_or_union_spec, struct_or_union, struct_decl_list, init_declarator_list, init_declarator, struct_decl, spec_qualifier_list, struct_declarator_list, struct_declarator_list, struct_declarator, enum_spec, enumerator_list, enumerator, declarator, direct_declarator, pointer, type_qualifier_list, param_list, param_decl, id_list, initializer, initializer_list, type_name, abstract_declarator, direct_abstract_declarator, stat, labeled_stat, exp_stat,compound_stat, stat_list, selection_stat, iteration_stat, jump_stat, exp assignment_exp, assignment_operator, conditional_exp, logical_or_exp, logical_and_exp, inclusive_or_exp, exclusive_or_exp, and_exp, equality_exp, relational_exp, shift_expression, additive_exp, mult_exp, cast_exp, unary_exp, unary_operator, postfix_exp, primary_exp, argument_exp_list, consts, int_const, char_const, float_const, id, string, enumeration_const, storage_const, type_const, qual_const, struct_const, enum_const, DEFINE, IF, ELSE, FOR, DO, WHILE, BREAK, SWITCH, CONTINUE, RETURN, CASE, DEFAULT, GOTO, SIZEOF, PUNC, or_const, and_const, eq_const, shift_const, rel_const, inc_const, point_const, HEADER}

T = set of terminals
= {All ASCII characters}

S = start symbol = program_unit

P = set of productions

```
program_unit                  -> HEADER program_unit
                                   | DEFINE primary_exp program_unit

                                   | translation_unit


translation_unit              -> external_decl

                                   | translation_unit external_decl


external_decl                 -> function_definition
                                   | decl

function_definition           -> decl_specs declarator decl_list compound_stat
                                   | declarator decl_list compound_stat
                                   | decl_specs declarator
compound_stat
                                   | declarator compound_stat
```

```
decl                        -> decl_specs init_declarator_list ';'

                                    | decl_specs ';'

decl_list                   -> decl
                                    | decl_list decl

decl_specs                  -> storage_class_spec decl_specs
                                    | storage_class_spec
                                    | type_spec decl_specs

                                    | type_spec

                                    | type_qualifier decl_specs
                                    | type_qualifier

storage_class_spec      -> storage_const

type_spec                   -> type_const

                                    | struct_or_union_spec
                                    | enum_spec

type_qualifier          -> qual_const

struct_or_union_spec  -> struct_or_union id '{' struct_decl_list '}' ';'
                                    | struct_or_union id

struct_or_union             -> struct_const

struct_decl_list        -> struct_decl
                                    | struct_decl_list struct_decl

init_declarator_list  -> init_declarator
                                    | init_declarator_list ',' init_declarator

init_declarator         -> declarator
                                    | declarator '=' initializer

struct_decl                 -> spec_qualifier_list struct_declarator_list ';'

spec_qualifier_list     -> type_spec spec_qualifier_list
                                    | type_spec
```

```
                                        | type_qualifier spec_qualifier_list
                                        | type_qualifier

struct_declarator_list          -> struct_declarator
                                        | struct_declarator_list ',' struct_declarator

struct_declarator                   -> declarator
                                        | declarator ':' conditional_exp
                                        | ':' conditional_exp

enum_spec                               -> enum_const id '{' enumerator_list '}'
                                        | enum_const '{' enumerator_list '}'
                                        | enum_const id

enumerator_list                         -> enumerator
                                        | enumerator_list ',' enumerator

enumerator                              -> id
                                        | id '=' conditional_exp

declarator                              -> pointer direct_declarator
                                        | direct_declarator

direct_declarator                   -> id

                                        | '(' declarator ')'

                                        | direct_declarator '[' conditional_exp ']'

                                        | direct_declarator '['    ']'
                                        | direct_declarator '(' param_list ')'

                                        | direct_declarator '(' id_list ')'

                                        | direct_declarator '('    ')'


pointer                                 -> '*' type_qualifier_list
                                        | '*'
                                        | '*' type_qualifier_list pointer
                                        | '*' pointer

type_qualifier_list                 -> type_qualifier
                                        | type_qualifier_list type_qualifier
```

```
param_list                      -> param_decl
                                        | param_list ',' param_decl

param_decl                      -> decl_specs declarator
                                        | decl_specs abstract_declarator
                                        | decl_specs

id_list                         -> id
                                        | id_list ',' id

initializer                     -> assignment_exp
                                        | '{' initializer_list '}'
                                        | '{' initializer_list ',' '}'

initializer_list        -> initializer
                                        | initializer_list ',' initializer

type_name                       -> spec_qualifier_list abstract_declarator
                                        | spec_qualifier_list

abstract_declarator         -> pointer
                                        | pointer direct_abstract_declarator
                                        |       direct_abstract_declarator

direct_abstract_declarator  -> '(' abstract_declarator ')'
                                        | direct_abstract_declarator '['
conditional_exp ']'

                                        | '[' conditional_exp ']'
                                        | direct_abstract_declarator '[' ']'
                                        | '[' ']'
                                        | direct_abstract_declarator '(' param_list ')'
                                        | '(' param_list ')'
                                        | direct_abstract_declarator '(' ')'
                                        | '(' ')'

stat                            -> labeled_stat

                                        | exp_stat

                                        | compound_stat

                                        | selection_stat
                                        | iteration_stat
                                        | jump_stat

labeled_stat                  -> id ':' stat
                                        | CASE int_const ':' stat
                                        | DEFAULT ':' stat

exp_stat                          -> exp ';'
                                        | ';'

compound_stat                     -> '{' decl_list stat_list '}'

                                        | '{' stat_list '}'

                                        | '{' decl_list     '}'

                                        | '{' '}'


stat_list                         -> stat

                                        | stat_list stat


selection_stat                -> IF '(' exp ')' stat
                    %prec "then"
                                        | IF '(' exp ')' stat ELSE stat
                                        | SWITCH '(' exp ')' stat

iteration_stat                -> WHILE '(' exp ')' stat
                                        | DO stat WHILE '(' exp ')' ';'
                                        | FOR '(' exp ';' exp ';' exp ')' stat
                                        | FOR '(' exp ';' exp ';'  ')' stat
                                        | FOR '(' exp ';' ';' exp ')' stat
                                        | FOR '(' exp ';' ';' ')' stat
                                        | FOR '(' ';' exp ';' exp ')' stat
                                        | FOR '(' ';' exp ';' ')' stat
                                        | FOR '(' ';' ';' exp ')' stat
                                        | FOR '(' ';' ';' ')' stat

jump_stat                         -> GOTO id ';'
                                        | CONTINUE ';'
                                        | BREAK ';'
                                        | RETURN exp ';'
                                        | RETURN ';'

exp                                     -> assignment_exp
                                        | exp ',' assignment_exp

assignment_exp                     -> conditional_exp
                                        | unary_exp assignment_operator
assignment_exp

assignment_operator           -> PUNC
                                        | '='

conditional_exp                    -> logical_or_exp
                                        | logical_or_exp '?' exp ':' conditional_exp

logical_or_exp                -> logical_and_exp
                                        | logical_or_exp or_const logical_and_exp

logical_and_exp                    -> inclusive_or_exp
                                        | logical_and_exp and_const
inclusive_or_exp

inclusive_or_exp              -> exclusive_or_exp
                                        | inclusive_or_exp '|' exclusive_or_exp

exclusive_or_exp              -> and_exp
                                        | exclusive_or_exp '^' and_exp

and_exp                                 -> equality_exp
                                        | and_exp '&' equality_exp

equality_exp                  -> relational_exp
                                        | equality_exp eq_const relational_exp

relational_exp                -> shift_expression
                                        | relational_exp '<' shift_expression
                                        | relational_exp '>' shift_expression
                                        | relational_exp rel_const shift_expression

shift_expression              -> additive_exp
                                        | shift_expression shift_const additive_exp

additive_exp                  -> mult_exp
                                        | additive_exp '+' mult_exp
```

```
                                   | additive_exp '-' mult_exp

mult_exp                       -> cast_exp
                                   | mult_exp '*' cast_exp
                                   | mult_exp '/' cast_exp
                                   | mult_exp '%' cast_exp

cast_exp                       -> unary_exp
                                   | '(' type_name ')' cast_exp

unary_exp                      -> postfix_exp
                                   | inc_const unary_exp
                                   | unary_operator cast_exp
                                   | SIZEOF unary_exp
                                   | SIZEOF '(' type_name ')'

unary_operator                 -> '&' | '*' | '+' | '-' | '~' | '!'

postfix_exp                    -> primary_exp

                                   | postfix_exp '[' exp ']'
                                   | postfix_exp '(' argument_exp_list ')'
                                   | postfix_exp '(' ')'
                                   | postfix_exp '.' id
                                   | postfix_exp point_const id
                                   | postfix_exp inc_const

primary_exp                    -> id

                                   | consts

                                   | string

                                   | '(' exp ')'

argument_exp_list          -> assignment_exp
                                   | argument_exp_list ',' assignment_exp

consts                         -> int_const

                                   | char_const
                                   | float_const
                                   | enumeration_const
```

```
int_const               -> [0-9]+

char_const              -> "'"."'"

float_const             -> [0-9]+"."[0-9]+

id                      -> [a-zA-z_][a-zA-z_0-9]*

string                  -> \".*\"

enum_const              -> "enum"

storage_const           -> "auto"
                        | "register"
                        | "static"
                        | "extern"
                        | "typedef"

type_const              -> "void"
                        |  "char"
                        |  "short"
                        |  "int"
                        |  "long"
                        |  "float"
                        |  "double"
                        |  "signed"
                        |  "unsigned"

qual_const              -> "const"
                        |  "volatile"

struct_const            -> "struct"
                        |  "union"

DEFINE                  -> "#define"[ ]+[a-zA-z_][a-zA-z_0-9]*

IF                      -> "if"

ELSE                    -> "else"

FOR                     -> "for"

DO                      -> "do"
```

```
WHILE                   -> "while"

BREAK                   -> "break"

SWITCH                  -> "switch"

CONTINUE                -> "continue"

RETURN                  -> "return"

CASE                    -> "case"

DEFAULT                 -> "default"

GOTO                    -> "goto"

SIZEOF                  -> "sizeof"

PUNC                    -> "*="
                    |  "/="
                    |  "+="
                    |  "%="
                    |  ">>="
                    |  "-="
                    |  "<<="
                    |  "&="
                    |  "^="
                    |  "|="

or_const                -> "||"

and_const               -> "&&"

eq_const                -> "=="
                    |  "!="

shift_const             -> ">>"
                    |  "<<"

rel_const               -> "<="
                    |  ">="

inc_const               -> "++"
                    |  "--"
```

```
point_const              -> "->"

HEADER                    -> "#include"[ ]+<[a-zA-z_][a-zA-z_0-9.]*>
```