L(G) = (S, P, Vn, Vt):

Vn = {<program>,<identifier>,<function>,<fun_return>, <params>, <body>, <data_type>, <letter>, <alpha_numeric>, <digit>, <var_init>, <instruction>, <var_unary>, <return>, <assign>, <operation>, <sign>, <value>, <number>, <string>, <char>, <assign_sign>, <float>, <integer>, <for>, <while>, <if>, <comparison_sign>, <for_end>, <loop_body>, <comb_sign>},

Vt = {function, void, int, float, string, a,b...z, A,B...Z, _, 0,1…9, boolean, true, false, ++, --, return, +, -, *, /, “, ., print, (,), }, {, break, =, ==, >=, <=, <, >, !, !=, +=, -=, for, ;, &&, ||},

P = {

<program> -> <function>+

<function> -> function <fun_return> <identifier> (<params>*) { <body> }

<fun_return> -> void | <data_type>

<data_type> -> int | float | string | boolean

<identifier> -> <letter><alpha_numeric>*

<alpha_numeric> -> <letter>|<digit>

<letter> -> a | b | ... | z | A | B | ... | Z |

<digit> -> 0 | 1 | … | 8 | 9

<params> -> <data_type> <identifier>

<body> -> <instruction>*

<instruction> -> <flow_control> | <print> | <assign> | <return> | <var_init> | <var_unary>;

<var_unary> -> <identifier> ++ | <identifier> --

<return> -> return <identifier>;

<assign> -> <identifier> = <identifier>; | <identifier> = <operation>+;

<operation> -> <identifier> <sign> <identifier>

<sign> -> + | - | * | /

<var_init> -> <data_type> <identifier>; | <data_type> <identifier> = <value>; | <data_type> <identifier> <assign_sign> <identifier>

<assign_sign> -> -= | +=

<value> -> true | false | <number> | <string>

<number> -> <float>|<integer>

<float> -> <digit>+ . <digit>+

<integer> -> <digit>+

<string> -> “<char>*”

<char> -> <letter>|<digit>

<print> -> print (<string>) | print(<string> + <identifier>);

<flow_control> -> <for> | <while> | <if>

<for> -> for(int <identifier> = <integer>; <identifier> <comparison_sign> <identifier>; <for_end>) { <body> }

<comparison_sign> -> > | < | >= | <= | == | !=

<for_end> -> <var_unary> | <assign>

<if> -> if(condition) { <body> } | if(condition) {<body>} else { <body> }

<while> -> while(<condition>){<loop_body>}

<loop_body> -> <body> break; | <body>

<condition> -> <identifier> <comparison_sign> <identifier> | <identifier> <comparison_sign> <number> | !<condition> |

<condition> -> <condition> <comb_sign> <condition>

<comb_sign> -> && | ||

}