SPACE [ \t]

NEWLINE [\n]

SINGLELINECOMMENT \/[/]+.\*

DIGIT [0-9]

LOWERCASE [a-z]

UPPERCASE [A-Z]

LEFTPARANTHESIS \(

RIGHTPARANTHESIS \)

LEFTCURLY \{

RIGHTCURLY \}

LEFTSQUARE \[

RIGHTSQUARE \]

DOT \.

SEMICOLON \;

COLON \:

COMMA \,

ASSIGN =

EQUALS ==

NOTEQUAL !=

AND \&

OR \|

IMPLIES \->

IFANDONLYIF <\->

NEGATION !

CONNECTIVE [{AND}{OR}{IMPLIES}{IFANDONLYIF}]

CONCATENATION \+

IF if

ELSE else

TRUE true

FALSE false

TRUTH\_VALUE {TRUE}|{FALSE}

WHILE while

STRING \"([^\\\"]|\\\"|\\\n|\\\\)\*\"

ALPHANUMERIC {LOWERCASE}|{UPPERCASE}|{DIGIT}

CONSTANT\_IDENTIFIER \".\*\"

CLASS\_IDENTIFIER {UPPERCASE}{ALPHANUMERIC}\*

FUNCTION func

CONSTANT const

VOID void

TRUTHVALUE truthvalue

RETURN\_TYPE {VOID}|{TRUTHVALUE}

VARIABLE var

INPUT input{COLON}

OUTPUT output{COLON}

CLASS class

RETURN return

IDENTIFIER {LOWERCASE}{ALPHANUMERIC}\*

%%

{CONSTANT\_IDENTIFIER} printf("<CONSTANT\_IDENTIFIER>");

{WHILE} printf("<WHILE>");

{IF} printf("<IF>");

{ELSE} printf("<ELSE>");

{EQUALS} printf("<EQUALS>");

{NOTEQUAL} printf("<NOTEQUAL>");

{CLASS} printf("<CLASS>");

{RETURN} printf("<RETURN>");

{LEFTPARANTHESIS} printf("<LEFTPARANTHESIS>");

{RIGHTPARANTHESIS} printf("<RIGHTPARANTHESIS>");

{LEFTCURLY} printf("<LEFTCURLY>");

{RIGHTCURLY} printf("<RIGHTCURLY>");

{INPUT} printf("<INPUT>");

{OUTPUT} printf("<OUTPUT>");

{VARIABLE} printf("<VARIABLE>");

{FUNCTION} printf("<FUNCTION>");

{CONSTANT} printf("<CONSTANT>");

{SINGLELINECOMMENT} printf("<SINGLELINECOMMENT>");

{CLASS\_IDENTIFIER} printf("<CLASS\_IDENTIFIER>");

{SPACE} printf(" ");

{ASSIGN} printf("<ASSIGN>");

{AND} printf("<AND>");

{OR} printf("<OR>");

{IMPLIES} printf("<IMPLIES>");

{NEGATION} printf("<NEGATION>");

{IFANDONLYIF} printf("<IDANDONLYIF>");

{TRUE} printf("<TRUE>");

{FALSE} printf("<FALSE>");

{RETURN\_TYPE} printf("<RETURN\_TYPE>");

{IDENTIFIER} printf("<IDENTIFIER>");

%%

int yywrap(){

return 1;

}

int main(void){

yylex();

return 0;

}

**Test1**

class Test1{

//Testing cont and var instantiation

const "const 1" = true;

const "const2" = false;

var var1 = "const1" & "const2";

}

**Test2**

class Test2{

//Testing function declarations and conditional statements

const "const1" = true;

const "const2" = false;

var var1 = "const1" | "const2";

var var2 = "const2" & var1;

func truthvalue funcname(var1, var2){

if((const1 & const2) == var1){

var2 = true;

}

else{

var2 = false;

}

return var2;

}

}

**Test3**

class Test3{

//testing function calls

var var1 = true;

var var2 = false;

func void funcname(var parameter){

parameter = false;

}

funcname(var1);

}

**Test4**

class Test4{

//testing connectives

const "const1" = true;

var p = false;

var a = "const1" & p;

var b = "const1" | p;

var c = "const1" -> p;

var d = "const1" <-> p;

var e = !"const1";

}

**Test5**

class Test5{

//testing variable reassignment and while loop and input-output

var p = true;

p = false;

while(p != true){

p = true;

}

p = input:

output:(p);

}