Name: Pradyumn R Pai
Roll No: 50
Class: CS7A

PROGRAM CODE

for.lex:

```
%{
#include "y.tab.h"
#include <string.h>
%}
%option noyywrap
%%
[0-9]+ { return INTEGER; }
for { return FOR; }
int|float|char { return TYPE; }
[a-zA-Z_][a-zA-Z0-9_]* {return IDENTIFIER;}
(==) { return RELATIONAL_OPERATOR; }
=|(\+=)|(-=)|(\*=)|(\/=)|(\|\|=)|(&&=)|(<<=)|(>>=)|(\^=)|(%=) { return ASSIGN; }
[+/%*\-] { return ARITHMETIC_OPERATOR; }
(<=)|(>=)|(!=)|[<>] { return RELATIONAL_OPERATOR; }
(\|\|)|(&&) { return LOGICAL_OPERATOR; }
(<<)|(>>)|[&|^] { return BITWISE_OPERATOR;}
\( { return LPAREN; }
\) { return RPAREN; }
\{ { return LCURLY; }
\} { return RCURLY; }
; { return SEMICOLON; }
, { return COMMA; }
! { return NOT; }
. { /*Skip remaining characters */ }
%%
```

for.y:

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
%}

%token IDENTIFIER
%token INTEGER
%token FOR
%token TYPE
%token LOGICAL_OPERATOR
%token RELATIONAL_OPERATOR
%token ARITHMETIC_OPERATOR
%token BITWISE_OPERATOR
%token LPAREN
%token RPAREN
%token LCURLY
%token RCURLY
%token SEMICOLON
%token COMMA
%token NOT
%token ASSIGN

%left LOGICAL_OPERATOR
%left RELATIONAL_OPERATOR
%left ARITHMETIC_OPERATOR
%left BITWISE_OPERATOR
%right NOT
%left LPAREN RPAREN
```

```
%%
//Grammar
program: /* empty */
    | program for_loop {printf("Valid\n");}
for_loop: FOR LPAREN init_statements SEMICOLON logic_opt SEMICOLON
update_stmts RPAREN stmt_block;
init_statements:  /* empty */
    | init_stmt_list;
logic_opt: /* empty */
    | logic_exp;
init_stmt_list: init_stmt
    | init_stmt_list COMMA init_stmt;
init_stmt: assignment_stmt
    | TYPE id_def;
update_stmts:  /* empty */
    | update_stmt_list;
update_stmt_list: update_stmt
    | update_stmt_list COMMA update_stmt;
update_stmt: assignment_stmt
    | arithmetic_exp;
stmt_block:LCURLY stmt_list RCURLY
    | stmt;
stmt_list: stmt
    | stmt_list stmt;
stmt: assignment_stmt SEMICOLON
    | for_loop
    | declaration_stmt SEMICOLON
    | arithmetic_exp SEMICOLON;
assignment_stmt: IDENTIFIER assignment_operator arithmetic_exp;
assignment_operator: ASSIGN
```

```
    ;
declaration_stmt : TYPE id_list
id_list : id_list COMMA id_def
    | id_def;
id_def: IDENTIFIER
    | assignment_stmt;
arithmetic_exp: arithmetic_exp ARITHMETIC_OPERATOR arithmetic_exp
    | arithmetic_exp BITWISE_OPERATOR arithmetic_exp
    | LPAREN arithmetic_exp RPAREN
    | IDENTIFIER
    | INTEGER;
logic_exp: arithmetic_exp RELATIONAL_OPERATOR arithmetic_exp
    | logic_exp LOGICAL_OPERATOR logic_exp
    | NOT logic_exp
    | LPAREN logic_exp RPAREN;

%%
void yyerror(char *s){
    printf("Error: %s\n",s);
}
int main() {
    yyparse();
    return 0;
}
int yywrap() {
    return 1;
}
```

OUTPUT:

```
for (int i=0;i<n;i+=1) {
    a = i*3;
    b = 6;
}
for (a=0,b=1,c=2;c<10;a=a+3,b=b-2,c=c+5) x = a*b*c;
for (int i=0;i<n;i=i+1){
    for (int j=0;j<m;j=j+2){
        x = x + i-j;
    }
}
for (;;) ;
for (;;;)
```

output:

```
Valid
Valid
Valid
Error: syntax error
```