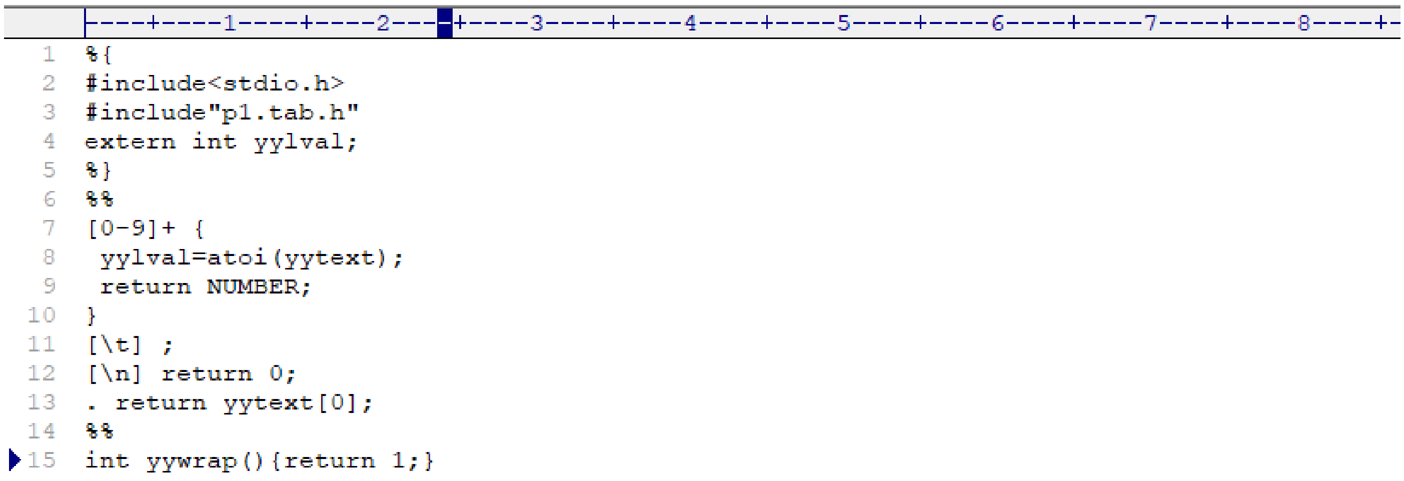
02/11/2021 **CD LAB 8** 2019103573

1)Write a yacc program to implement arithmetic operators.

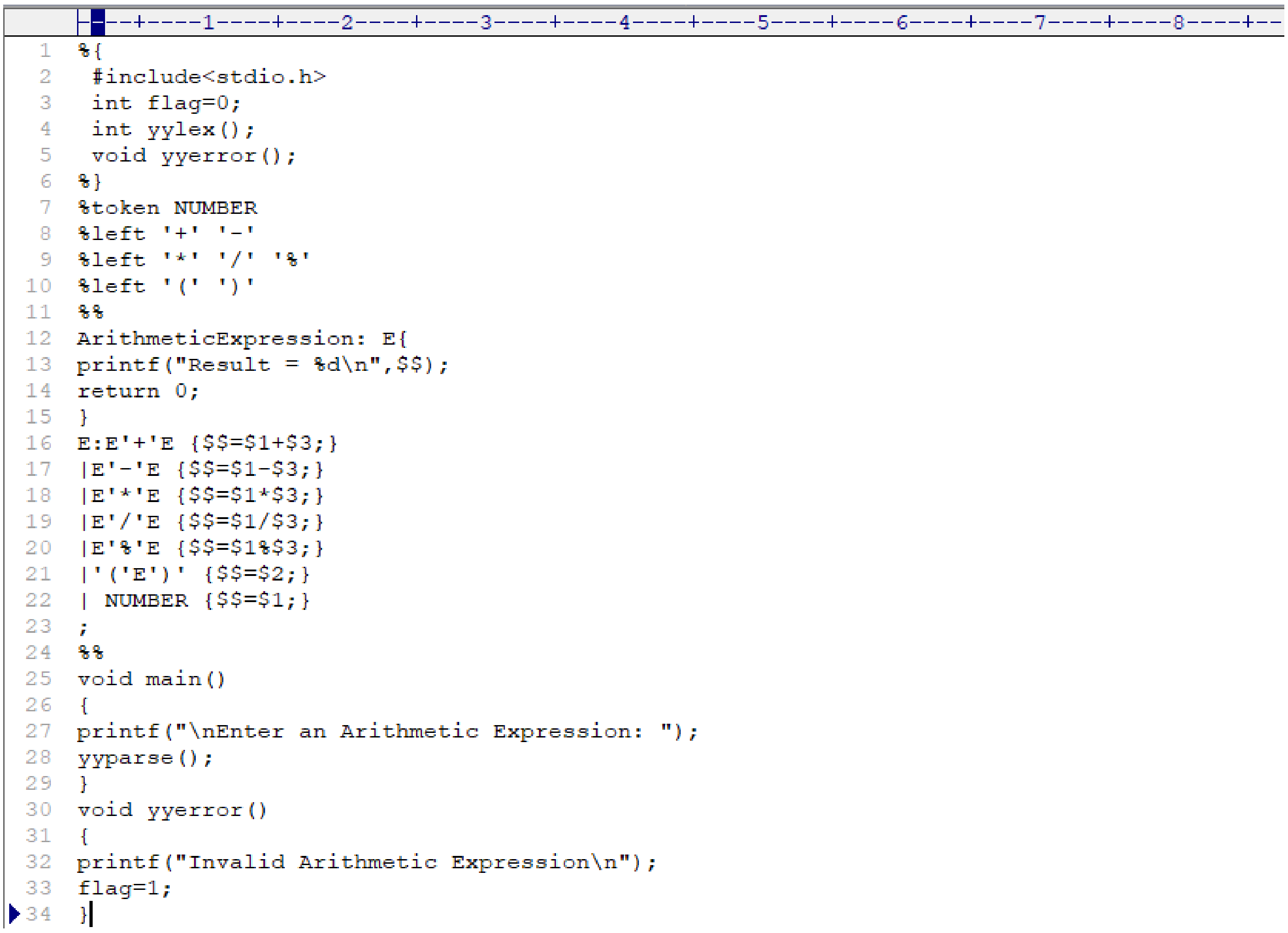
CODE :-

lex



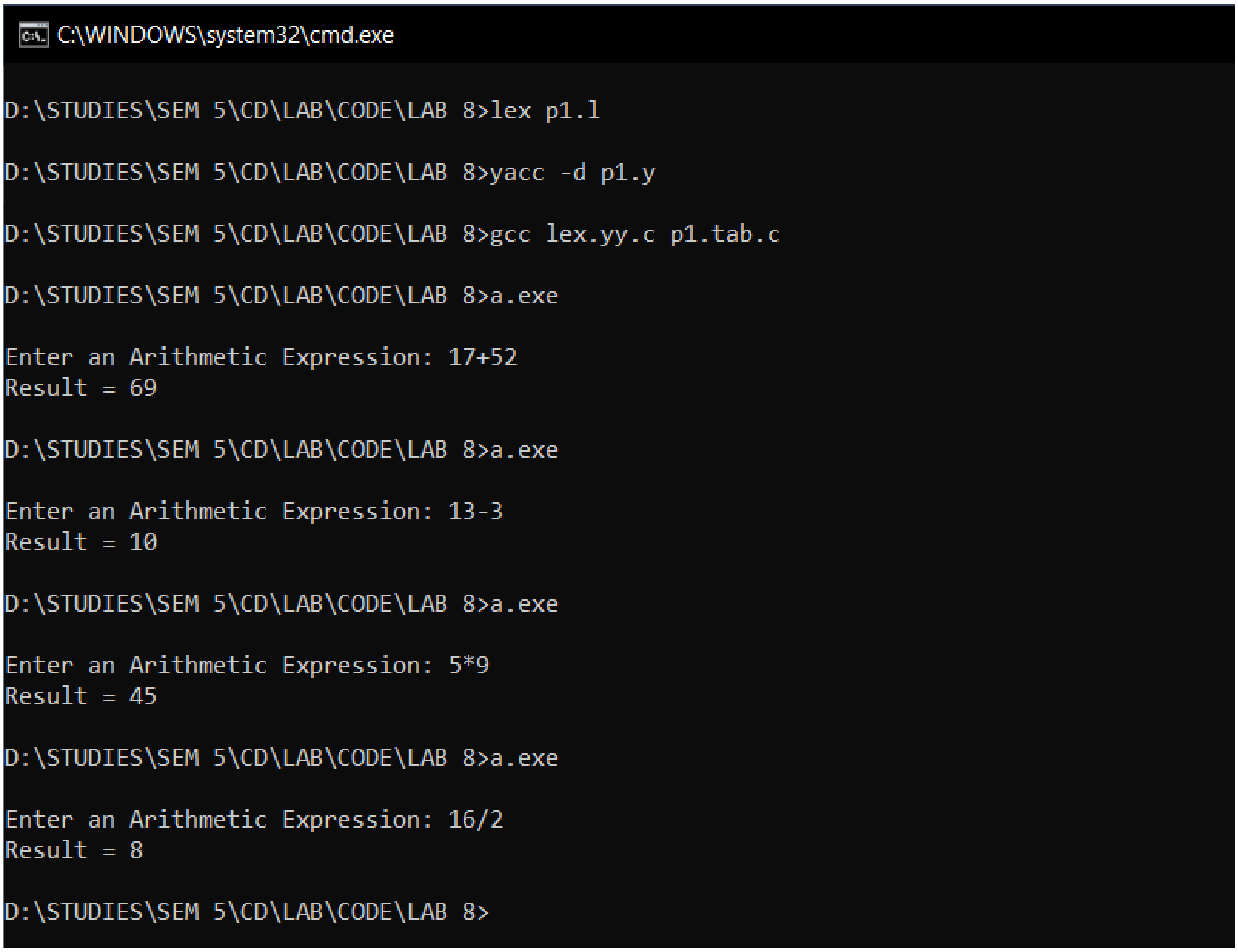
```
%{
#include<stdio.h>
#include"p1.tab.h"
extern int yylval;
%}
%%
[0-9]+ {
 yylval=atoi(yytext);
 return NUMBER;
}
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap(){return 1;}
```

yacc



```
%{
 #include<stdio.h>
 int flag=0;
 int yylex();
 void yyerror();
%}
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E{
printf("Result = %d\n",$$);
return 0;
}
E:E'+'E {$$=$1+$3;}
|E'-'E {$$=$1-$3;}
|E'*'E {$$=$1*$3;}
|E'/'E {$$=$1/$3;}
|E'%'E {$$=$1%$3;}
|'('E')' {$$=$2;}
| NUMBER {$$=$1;}
;
%%
void main()
{
printf("\nEnter an Arithmetic Expression: ");
yyparse();
}
void yyerror()
{
printf("Invalid Arithmetic Expression\n");
flag=1;
}
```

OUTPUT :-



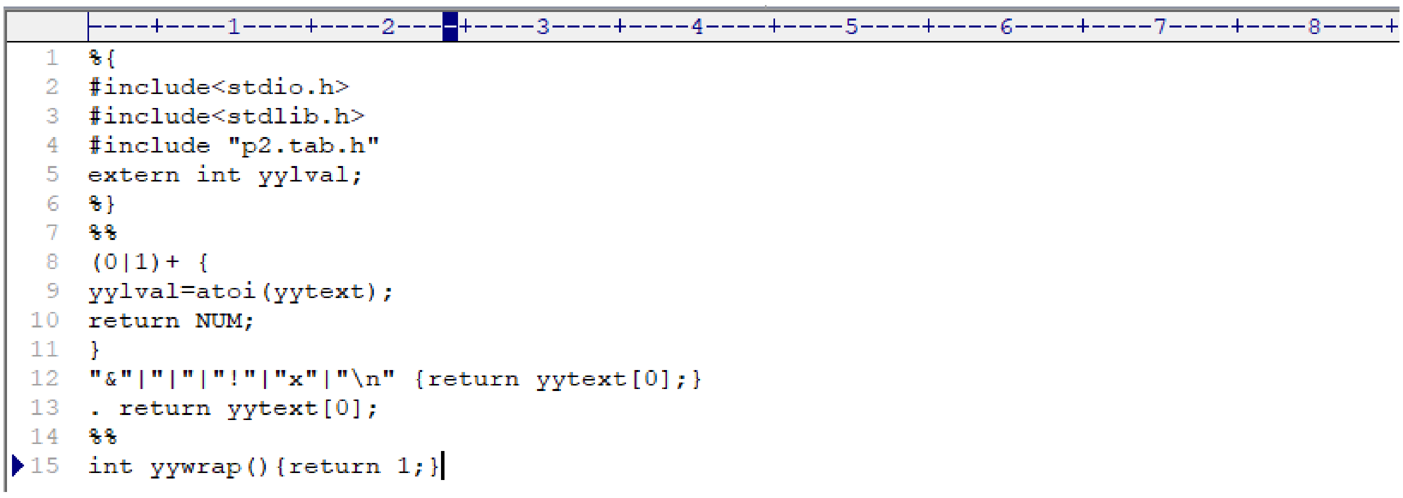
C:\WINDOWS\system32\cmd.exe

```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>lex p1.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>yacc -d p1.y

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>gcc lex.yy.c p1.tab.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter an Arithmetic Expression: 17+52
Result = 69

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter an Arithmetic Expression: 13-3
Result = 10

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter an Arithmetic Expression: 5*9
Result = 45

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter an Arithmetic Expression: 16/2
Result = 8

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>
```

2) Write a yacc program to implement logical operators.

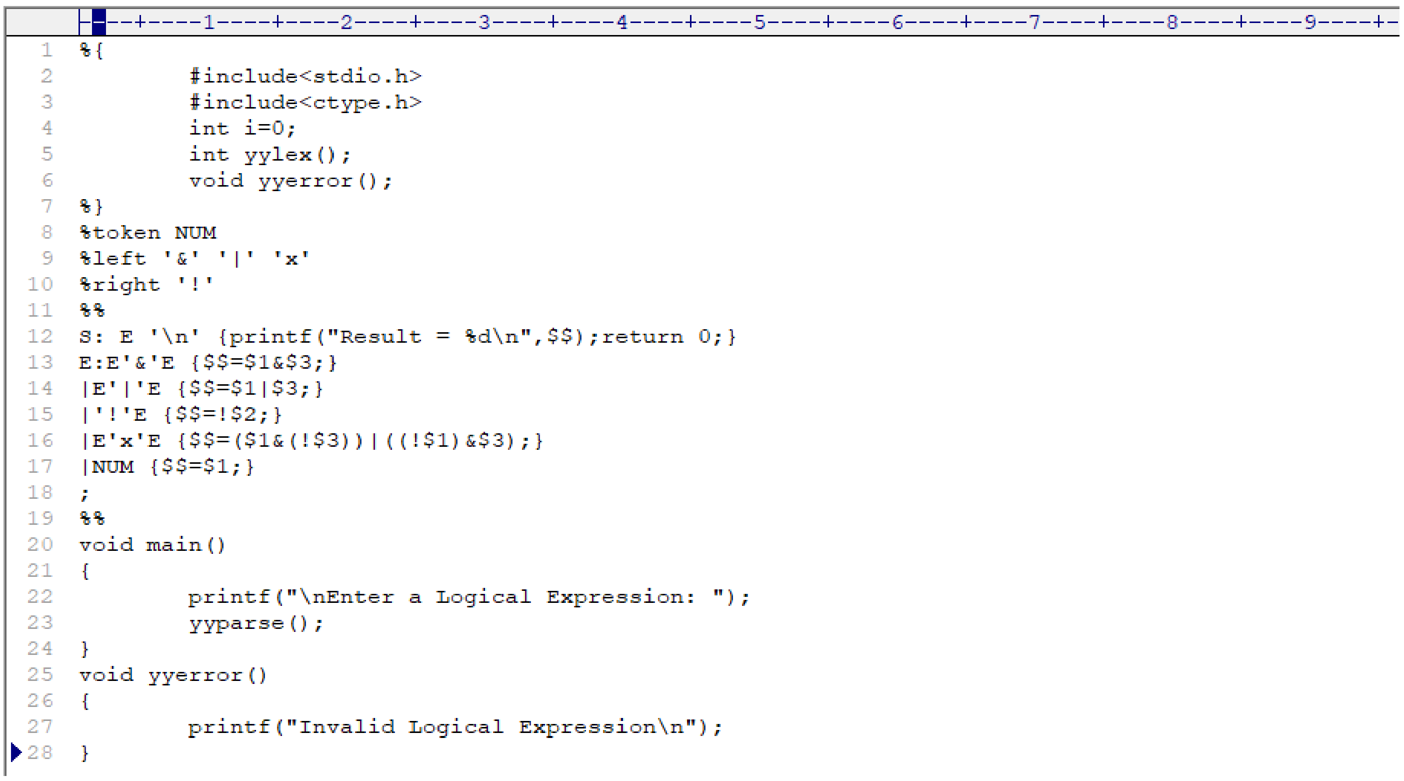
CODE :-

lex



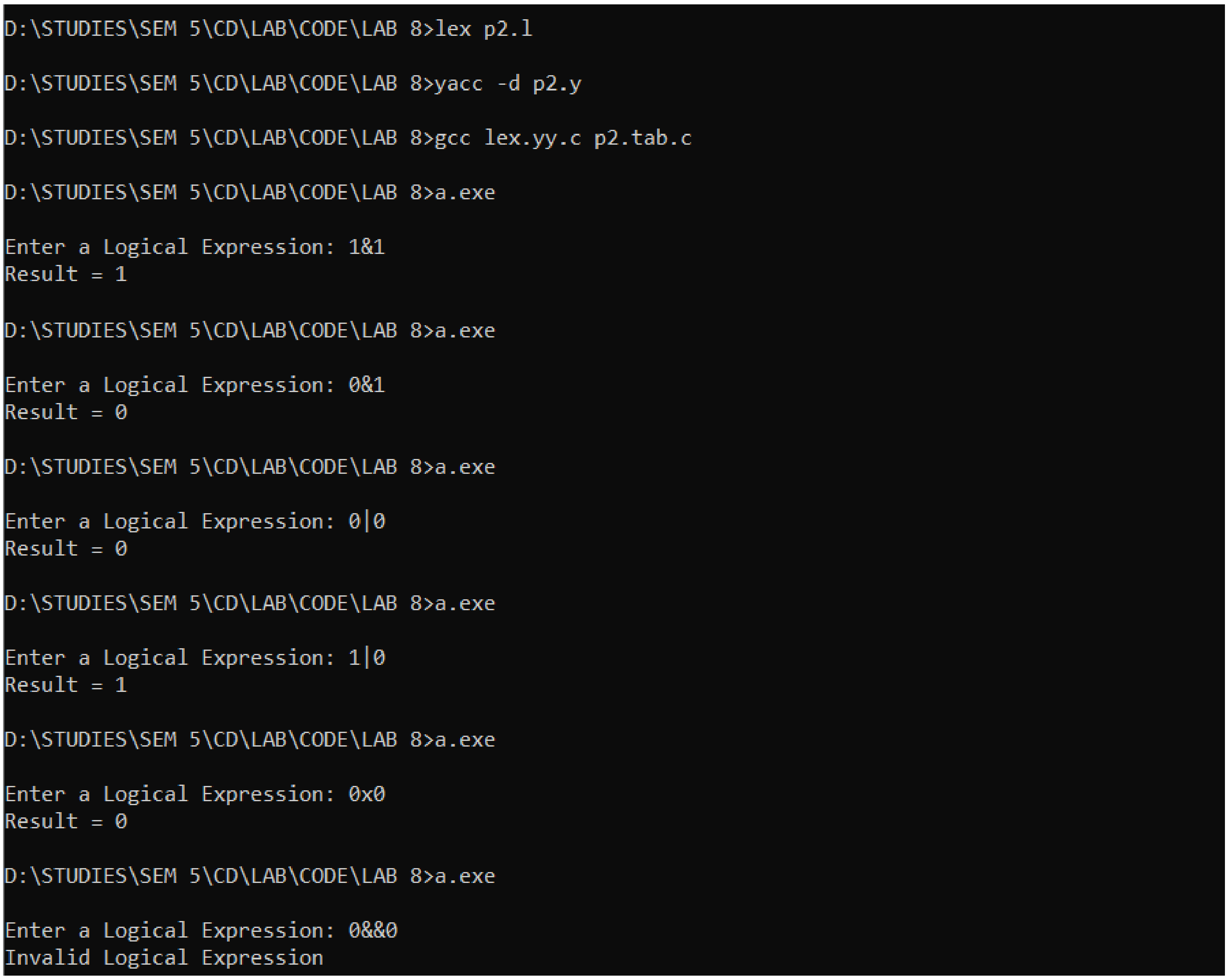
```
%{
#include<stdio.h>
#include<stdlib.h>
#include "p2.tab.h"
extern int yylval;
%}
%%
(0|1)+ {
yylval=atoi(yytext);
return NUM;
}
"&"|"|"|"!"|"x"|"\n" {return yytext[0];}
. return yytext[0];
%%
int yywrap(){return 1;}
```

yacc



```
%{
        #include<stdio.h>
        #include<ctype.h>
        int i=0;
        int yylex();
        void yyerror();
%}
%token NUM
%left '&' '|' 'x'
%right '!'
%%
S: E '\n' {printf("Result = %d\n",$$);return 0;}
E:E'&'E {$$=$1&$3;}
|E'|'E {$$=$1|$3;}
|'!'E {$$=!$2;}
|E'x'E {$$=($1&(!$3))|((!$1)&$3);}
|NUM {$$=$1;}
;
%%
void main()
{
        printf("\nEnter a Logical Expression: ");
        yyparse();
}
void yyerror()
{
        printf("Invalid Logical Expression\n");
}
```

OUTPUT :-

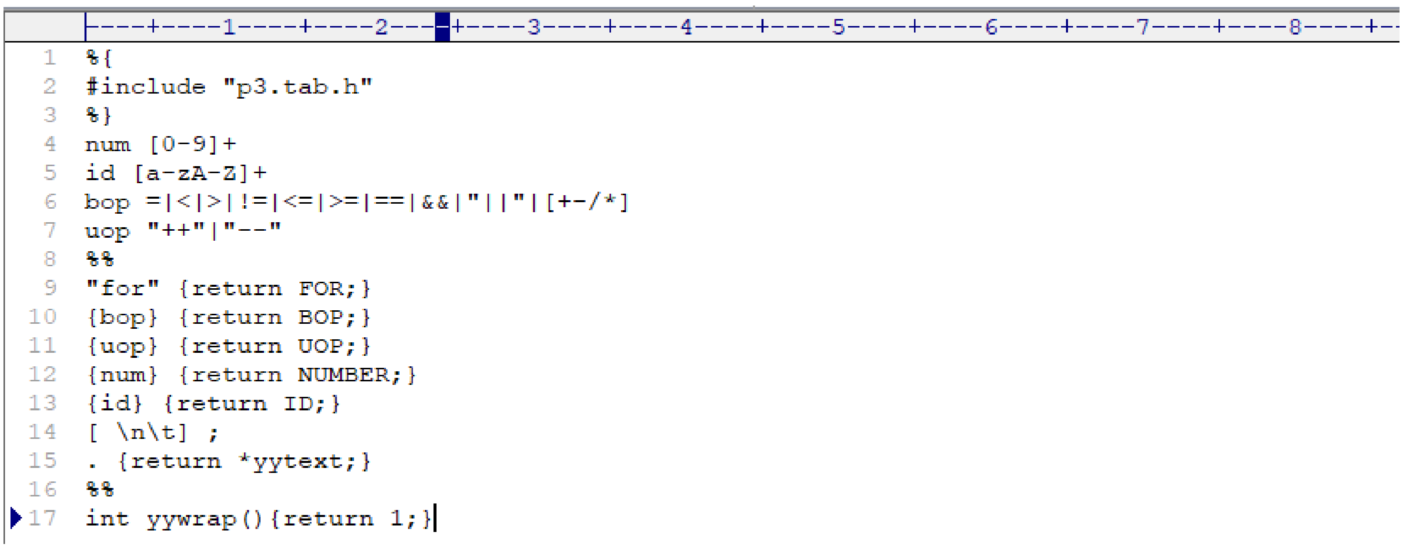


```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>lex p2.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>yacc -d p2.y

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>gcc lex.yy.c p2.tab.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter a Logical Expression: 1&1
Result = 1

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter a Logical Expression: 0&1
Result = 0

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter a Logical Expression: 0|0
Result = 0

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter a Logical Expression: 1|0
Result = 1

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter a Logical Expression: 0x0
Result = 0

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter a Logical Expression: 0&&0
Invalid Logical Expression
```

3) Write a yacc program to check the syntax of for expression.

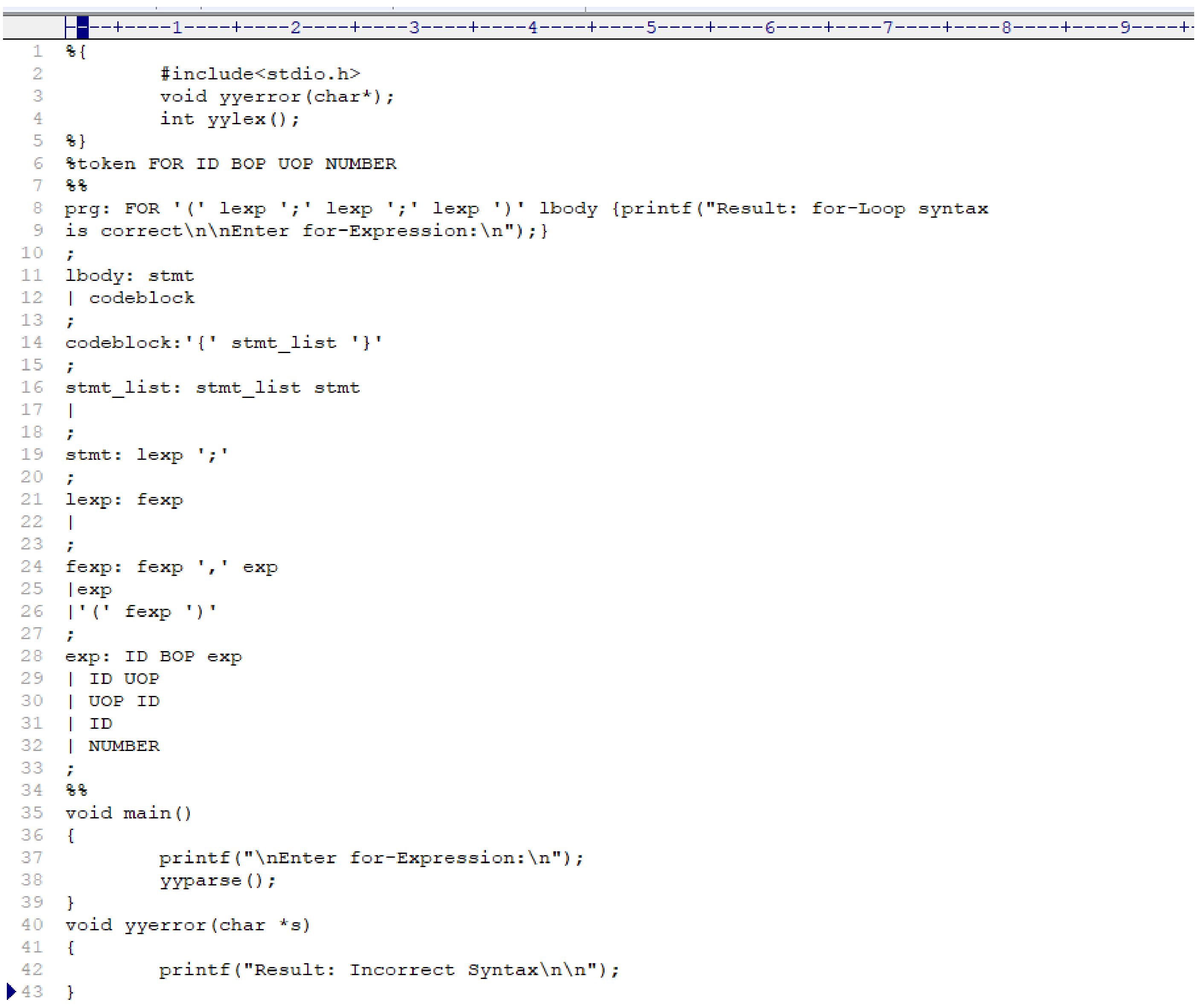
CODE :-

lex



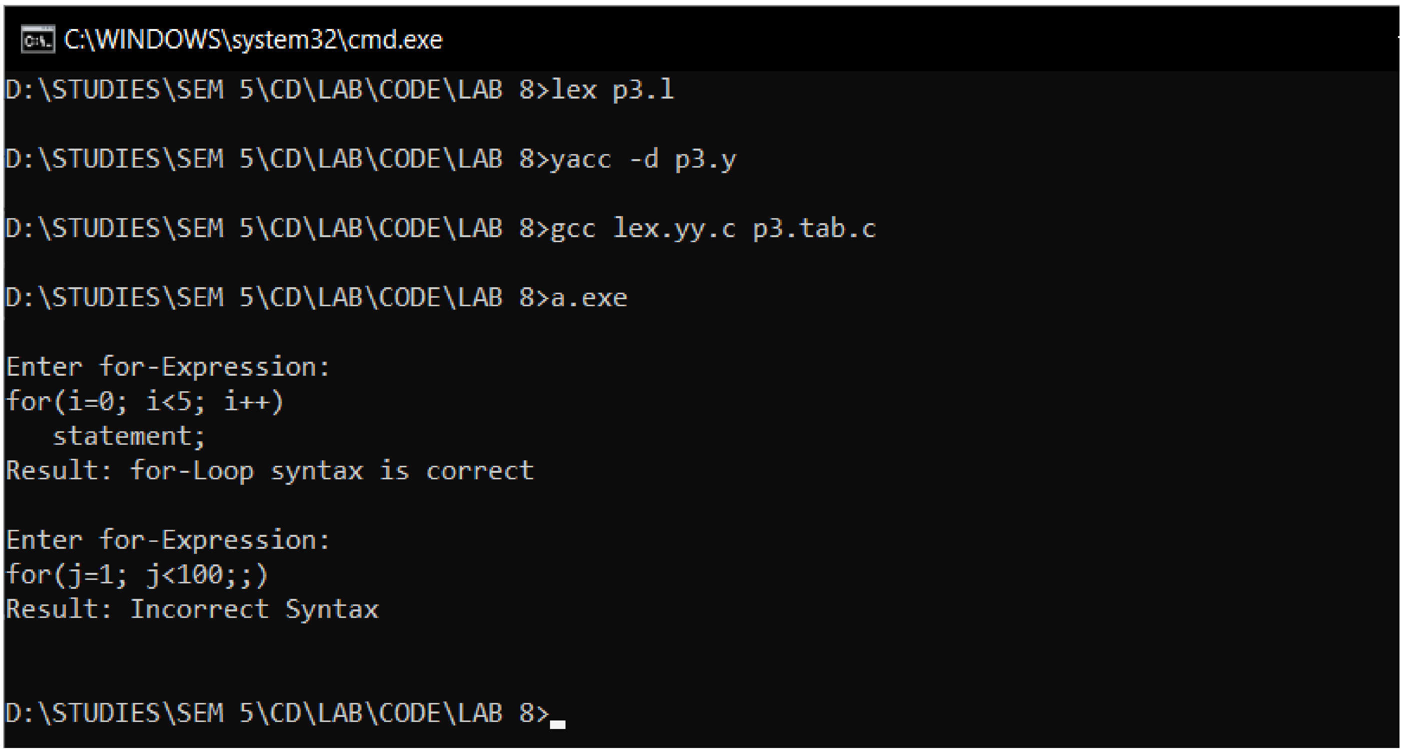
```
%{
#include "p3.tab.h"
%}
num [0-9]+
id [a-zA-Z]+
bop =|<|>|!=|<=|>=|==|&&|"||"|[+-/*]
uop "++"|"--"
%%
"for" {return FOR;}
{bop} {return BOP;}
{uop} {return UOP;}
{num} {return NUMBER;}
{id} {return ID;}
[ \n\t] ;
. {return *yytext;}
%%
int yywrap(){return 1;}
```

yacc



```
%{
        #include<stdio.h>
        void yyerror(char*);
        int yylex();
%}
%token FOR ID BOP UOP NUMBER
%%
prg: FOR '(' lexp ';' lexp ';' lexp ')' lbody {printf("Result: for-Loop syntax
is correct\n\nEnter for-Expression:\n");}
;
lbody: stmt
| codeblock
;
codeblock:'{' stmt_list '}'
;
stmt_list: stmt_list stmt
|
;
stmt: lexp ';'
;
lexp: fexp
|
;
fexp: fexp ',' exp
|exp
|'(' fexp ')'
;
exp: ID BOP exp
| ID UOP
| UOP ID
| ID
| NUMBER
;
%%
void main()
{
        printf("\nEnter for-Expression:\n");
        yyparse();
}
void yyerror(char *s)
{
        printf("Result: Incorrect Syntax\n\n");
}
```

OUTPUT


C:\WINDOWS\system32\cmd.exe

```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>lex p3.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>yacc -d p3.y

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>gcc lex.yy.c p3.tab.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>a.exe

Enter for-Expression:
for(i=0; i<5; i++)
   statement;
Result: for-Loop syntax is correct

Enter for-Expression:
for(j=1; j<100;;)
Result: Incorrect Syntax


D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 8>
```