CSB 353: Compiler Design

LAB 9

Submitted By:

Name: PREM KUMAR

Roll No: 191210037

Branch: CSE

Semester: 6 th

Submitted To: Dr. Shelly Sachdeva

Department of Computer Science and Engineering

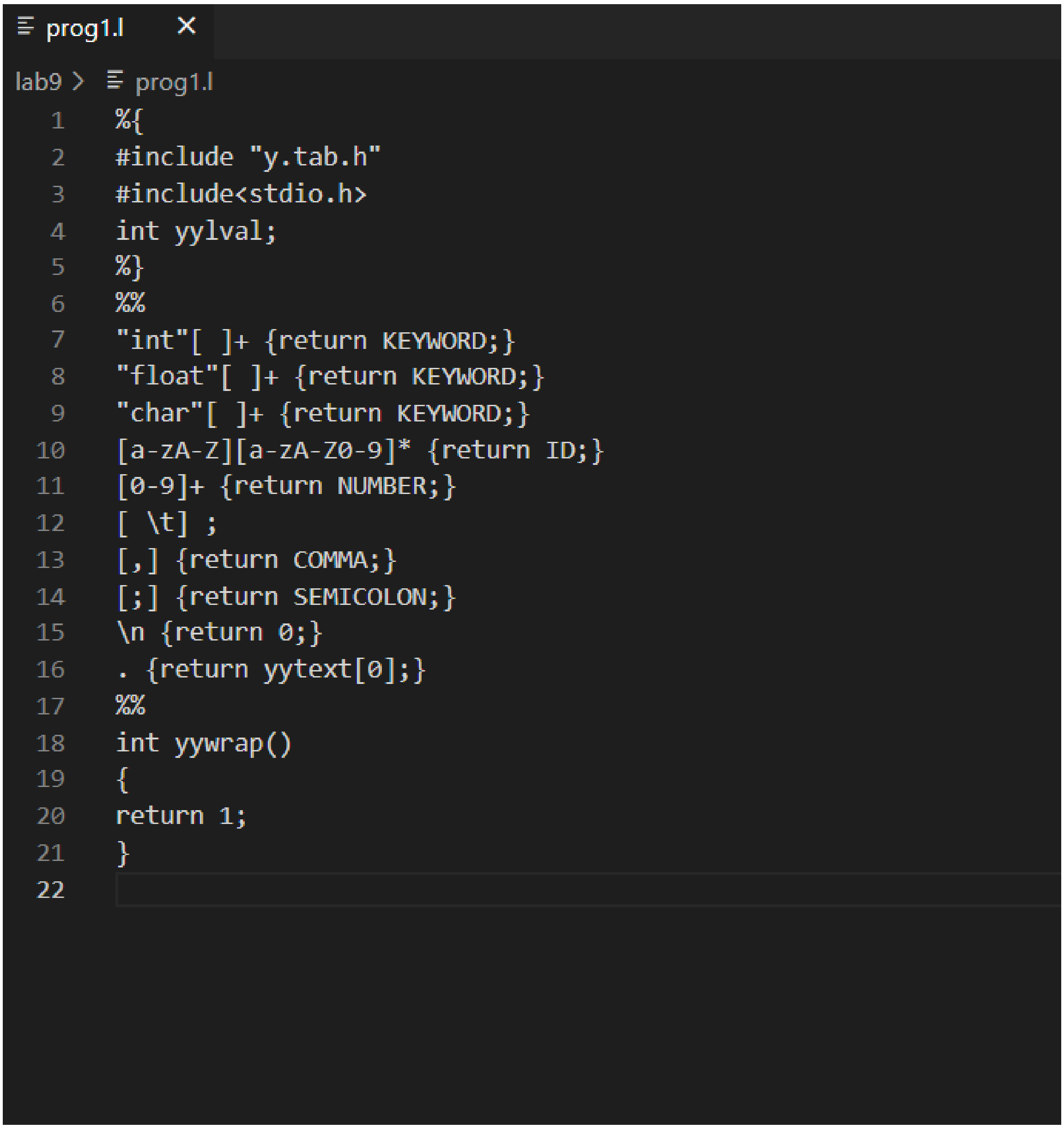
NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2019-2023

Ques 1. Design a grammar for a declarative statement for C program. Further, write a Yacc program to check if entered statement is a valid declarative statement according to the grammar generated.

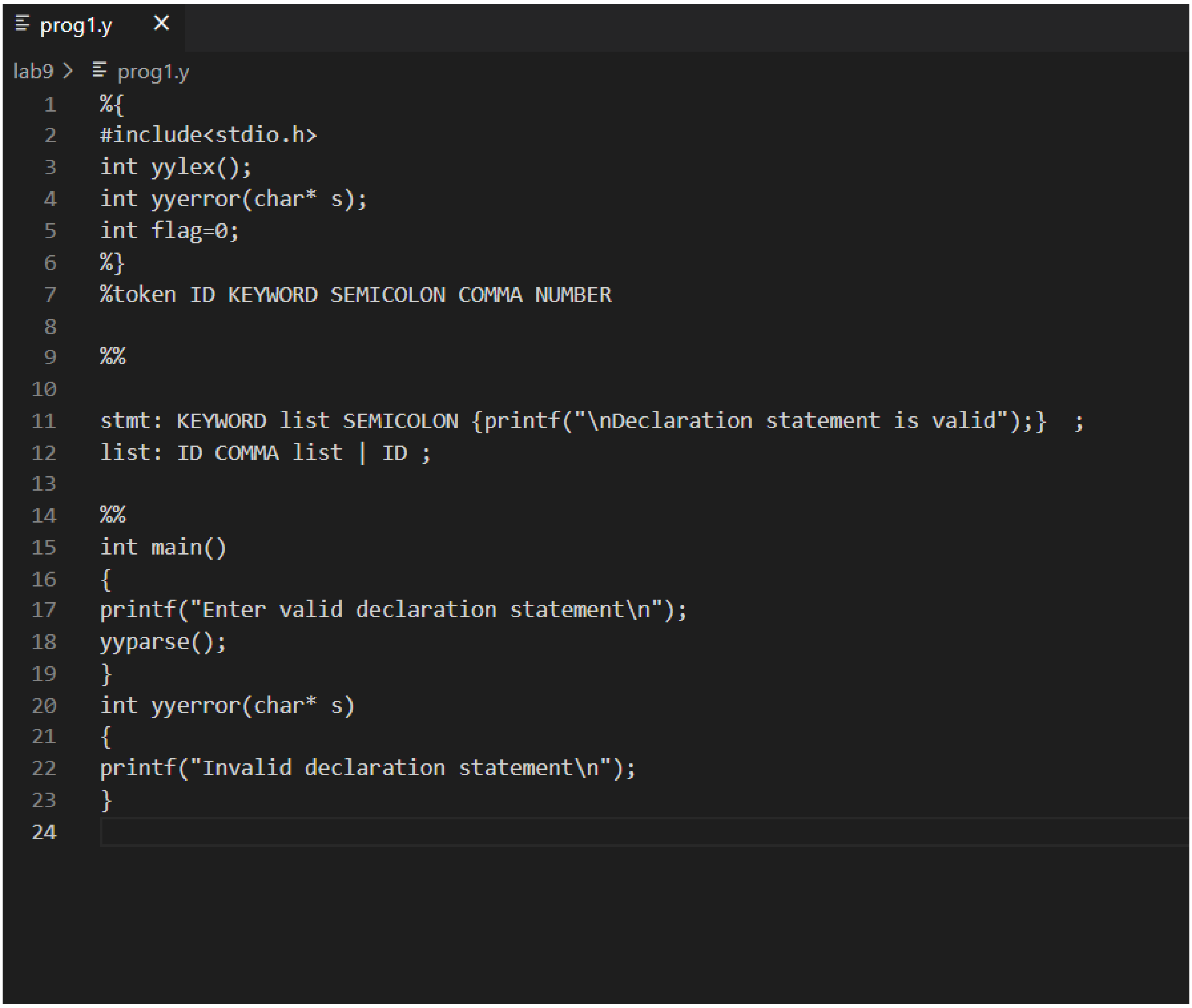
Code:

- prog1.l (Lex File)



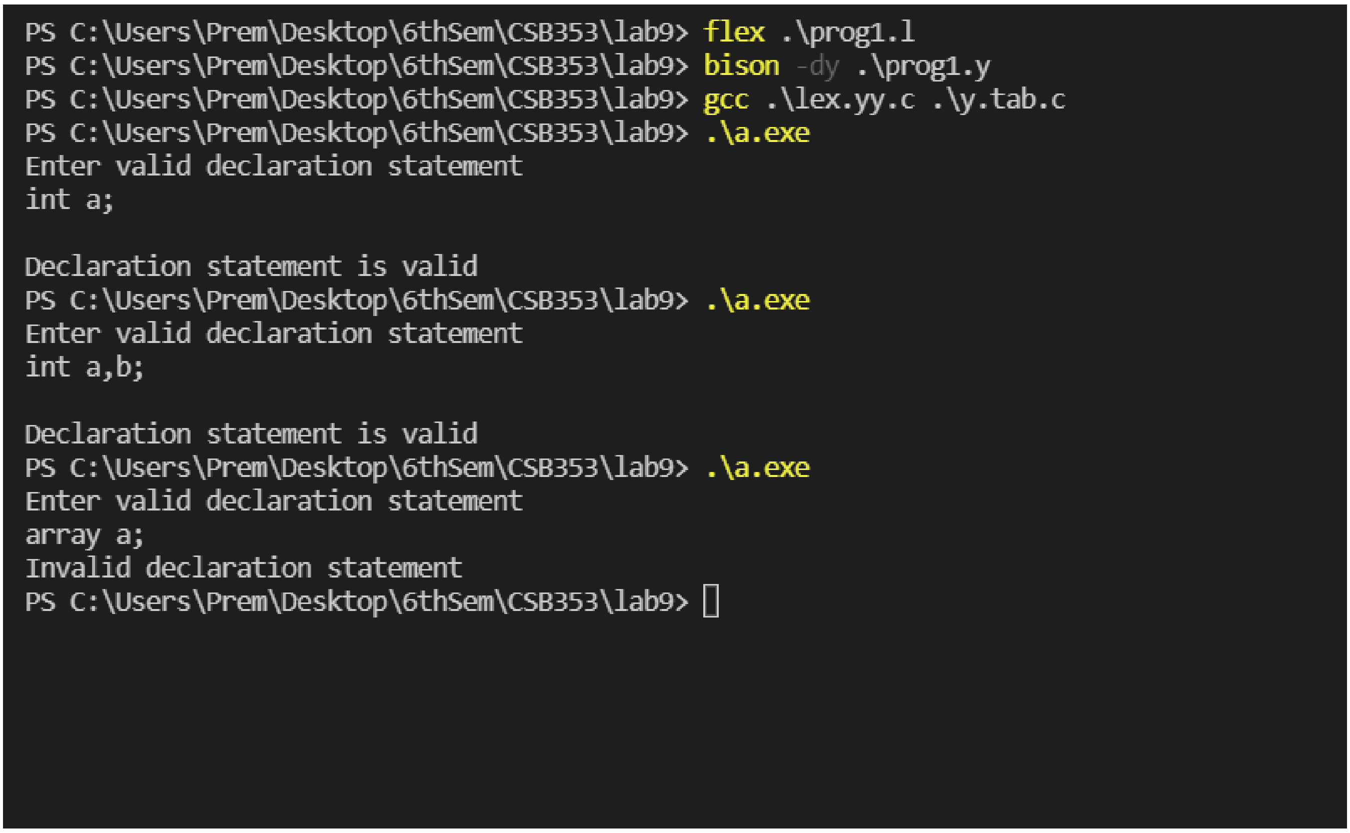
```
%{
#include "y.tab.h"
#include<stdio.h>
int yylval;
%}
%%
"int"[ ]+ {return KEYWORD;}
"float"[ ]+ {return KEYWORD;}
"char"[ ]+ {return KEYWORD;}
[a-zA-Z][a-zA-Z0-9]* {return ID;}
[0-9]+ {return NUMBER;}
[ \t] ;
[,] {return COMMA;}
[;] {return SEMICOLON;}
\n {return 0;}
. {return yytext[0];}
%%
int yywrap()
{
return 1;
}

```

- prog1.y (Yacc File)



```
%{
#include<stdio.h>
int yylex();
int yyerror(char* s);
int flag=0;
%}
%token ID KEYWORD SEMICOLON COMMA NUMBER

%%

stmt: KEYWORD list SEMICOLON {printf("\nDeclaration statement is valid");}  ;
list: ID COMMA list | ID ;

%%
int main()
{
printf("Enter valid declaration statement\n");
yyparse();
}
int yyerror(char* s)
{
printf("Invalid declaration statement\n");
}

```

Output:

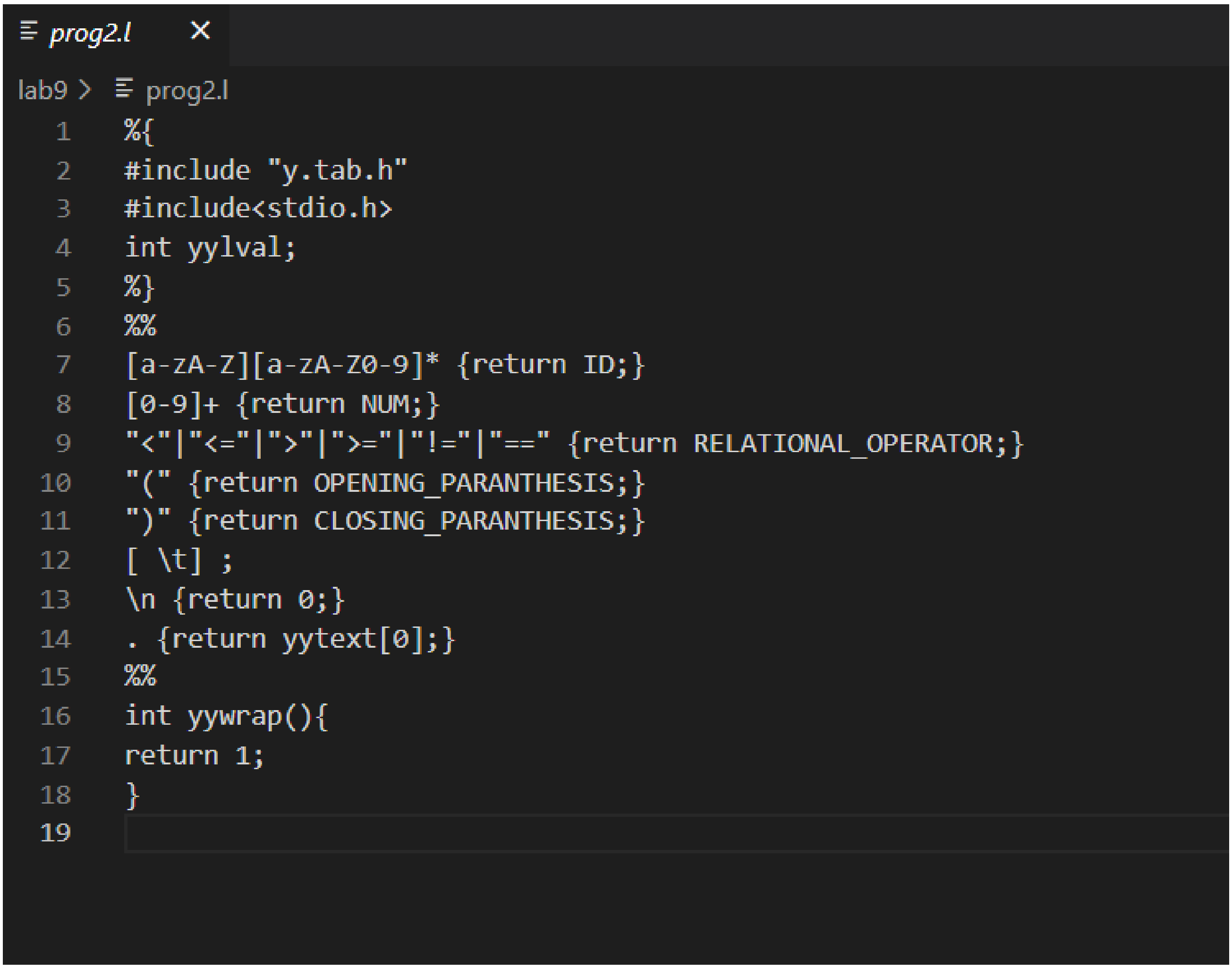


```
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> flex .\prog1.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> bison -dy .\prog1.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid declaration statement
int a;

Declaration statement is valid
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid declaration statement
int a,b;

Declaration statement is valid
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid declaration statement
array a;
Invalid declaration statement
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> 
```

Ques 2. Design a grammar for a relational expression of C language. Further, write a Yacc program to check if entered statement is a valid relational expression according to the grammar generated.

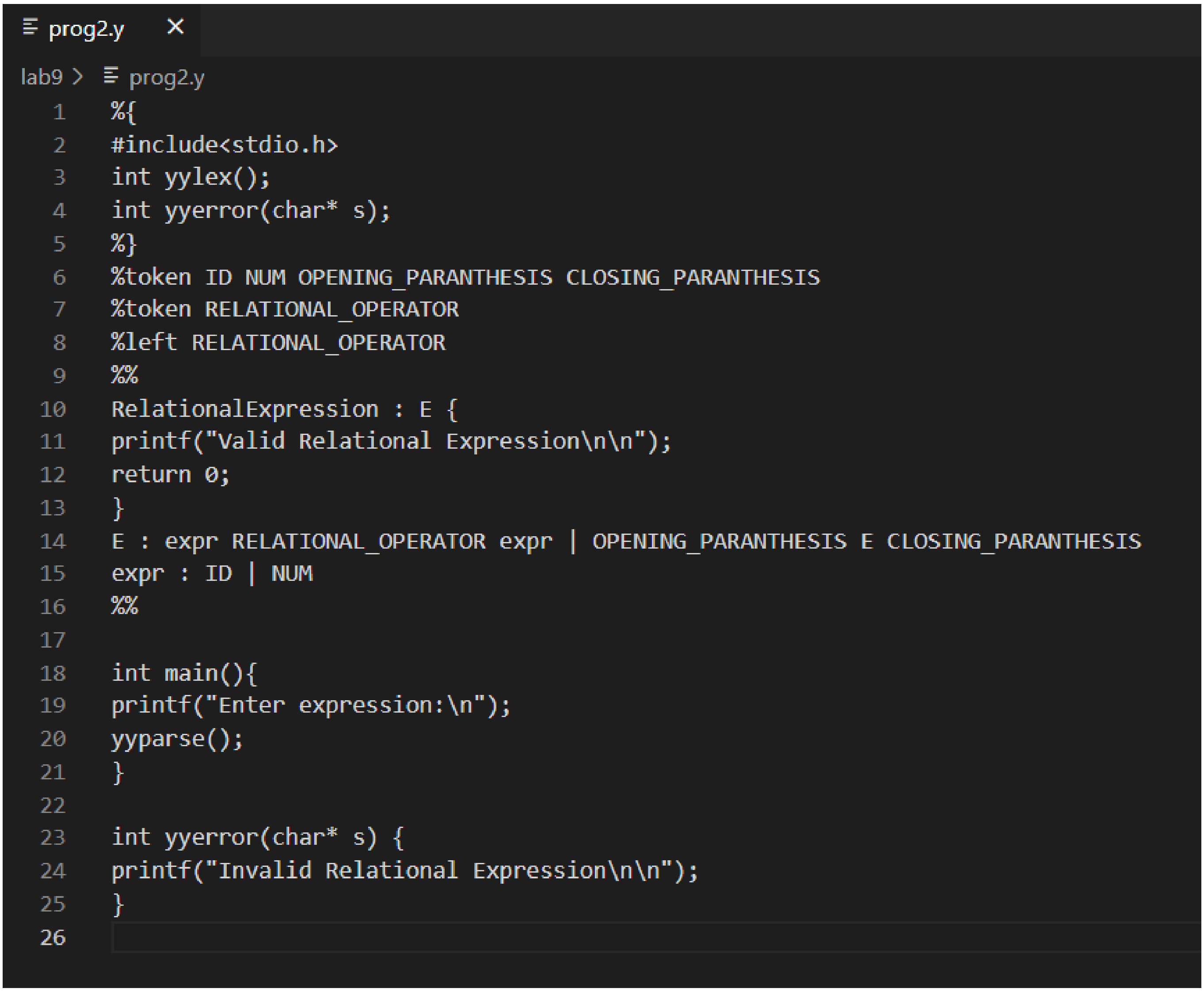
Code:

- prog2.l (Lex File)



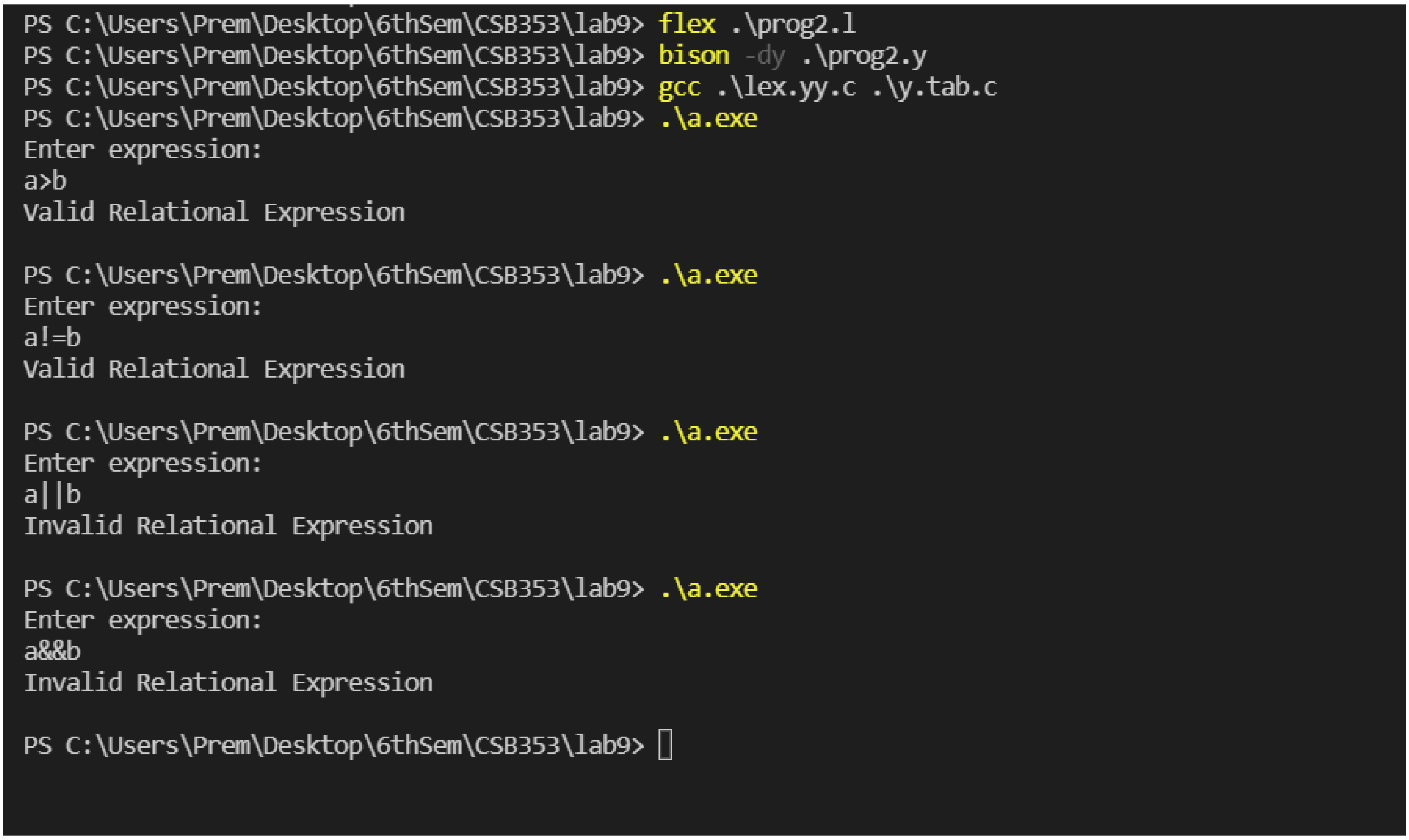
```
%{
#include "y.tab.h"
#include<stdio.h>
int yylval;
%}
%%
[a-zA-Z][a-zA-Z0-9]* {return ID;}
[0-9]+ {return NUM;}
"<"|"<="|">"|">="|"!="|"==" {return RELATIONAL_OPERATOR;}
"(" {return OPENING_PARANTHESIS;}
")" {return CLOSING_PARANTHESIS;}
[ \t] ;
\n {return 0;}
. {return yytext[0];}
%%
int yywrap(){
return 1;
}

```

- prog2.y (Yacc File)



```
%{
#include<stdio.h>
int yylex();
int yyerror(char* s);
%}
%token ID NUM OPENING_PARANTHESIS CLOSING_PARANTHESIS
%token RELATIONAL_OPERATOR
%left RELATIONAL_OPERATOR
%%
RelationalExpression : E {
printf("Valid Relational Expression\n\n");
return 0;
}
E : expr RELATIONAL_OPERATOR expr | OPENING_PARANTHESIS E CLOSING_PARANTHESIS
expr : ID | NUM
%%

int main(){
printf("Enter expression:\n");
yyparse();
}

int yyerror(char* s) {
printf("Invalid Relational Expression\n\n");
}

```

Output:

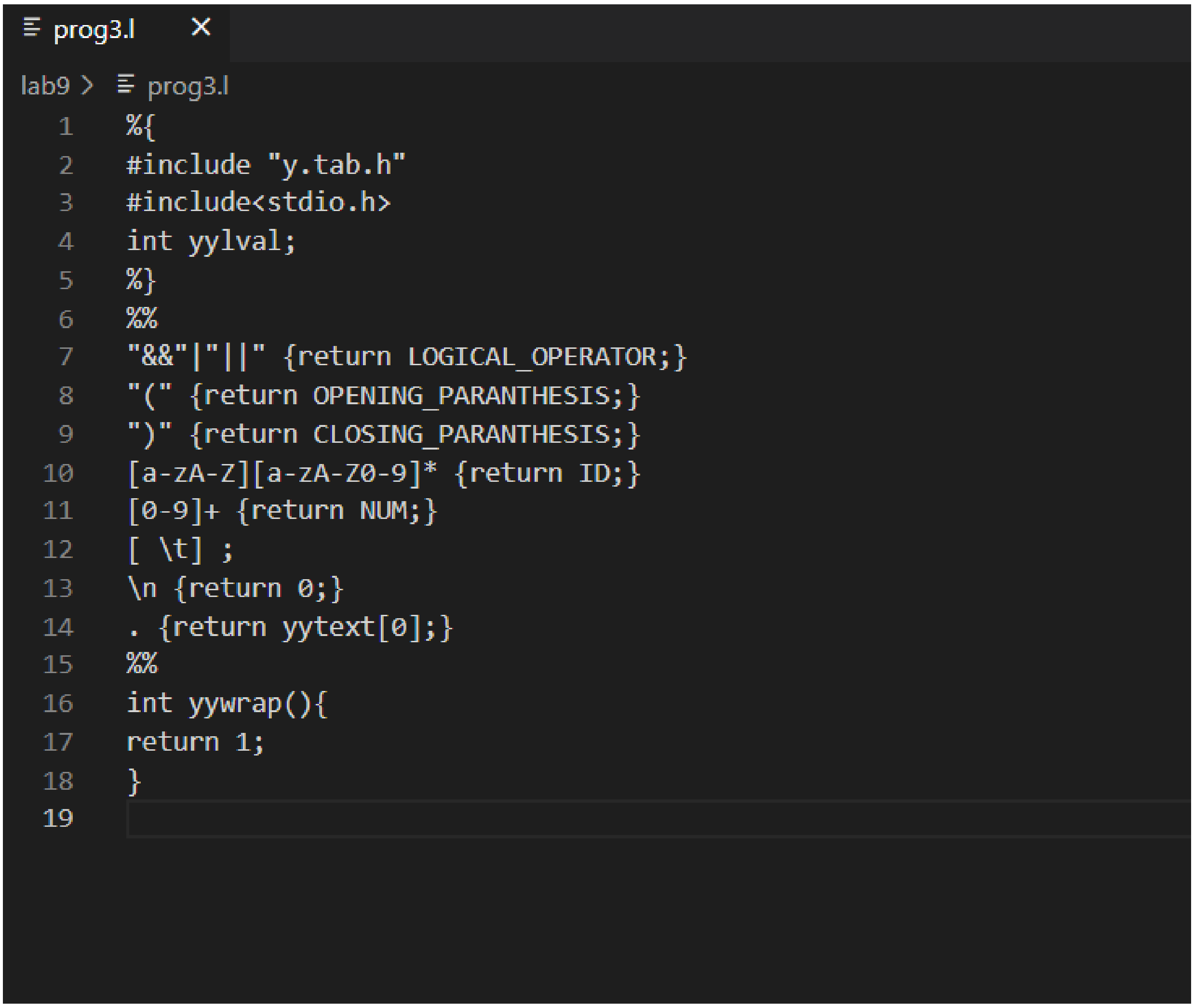


```
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> flex .\prog2.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> bison -dy .\prog2.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter expression:
a>b
Valid Relational Expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter expression:
a!=b
Valid Relational Expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter expression:
a||b
Invalid Relational Expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter expression:
a&&b
Invalid Relational Expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> 
```

Ques 3. Design a grammar for a logical expression of C langauge. Further, write a Yacc program to check if entered statement is a valid logical expression according to the grammar generated.

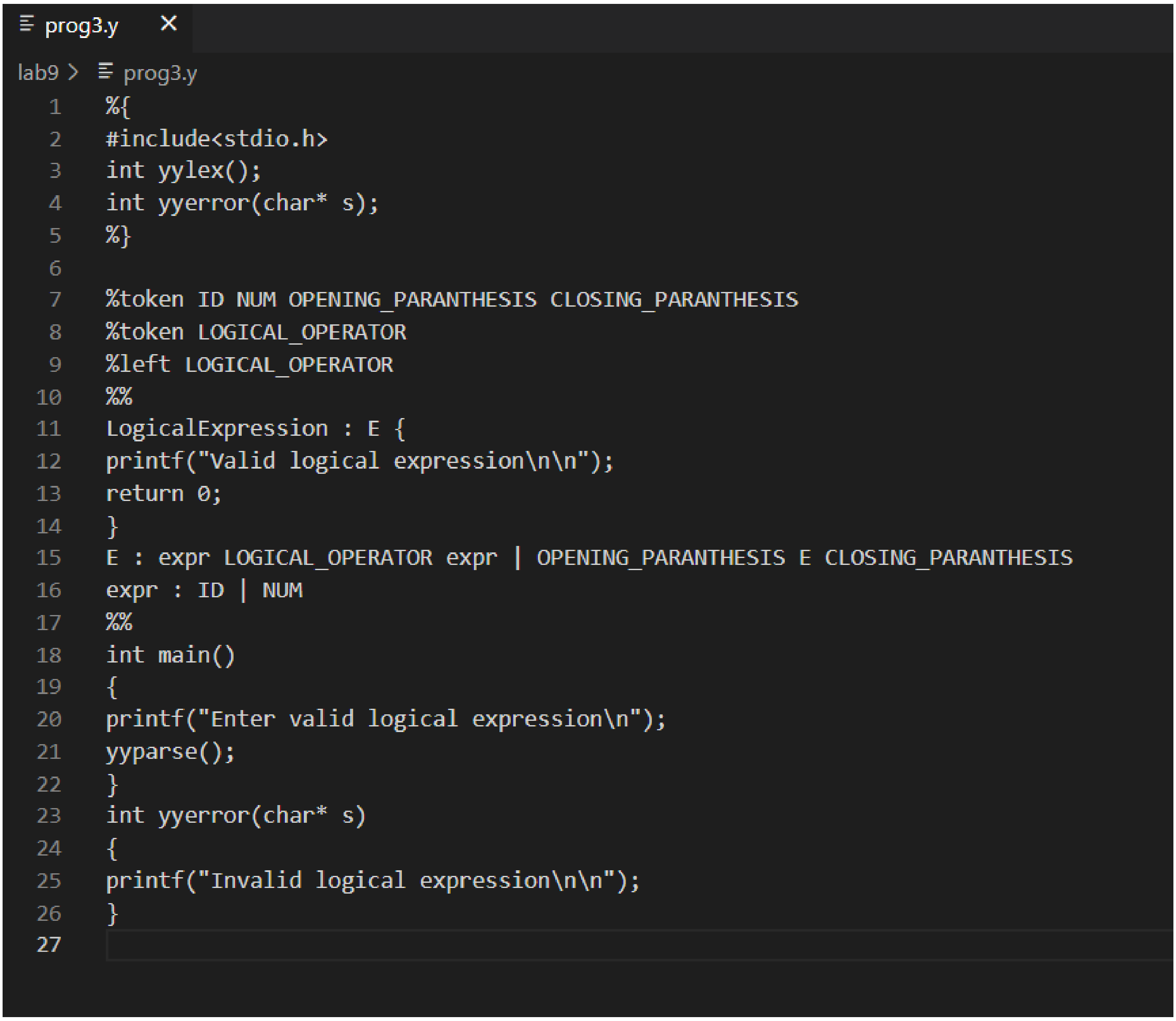
Code:

- prog3.l (Lex File)



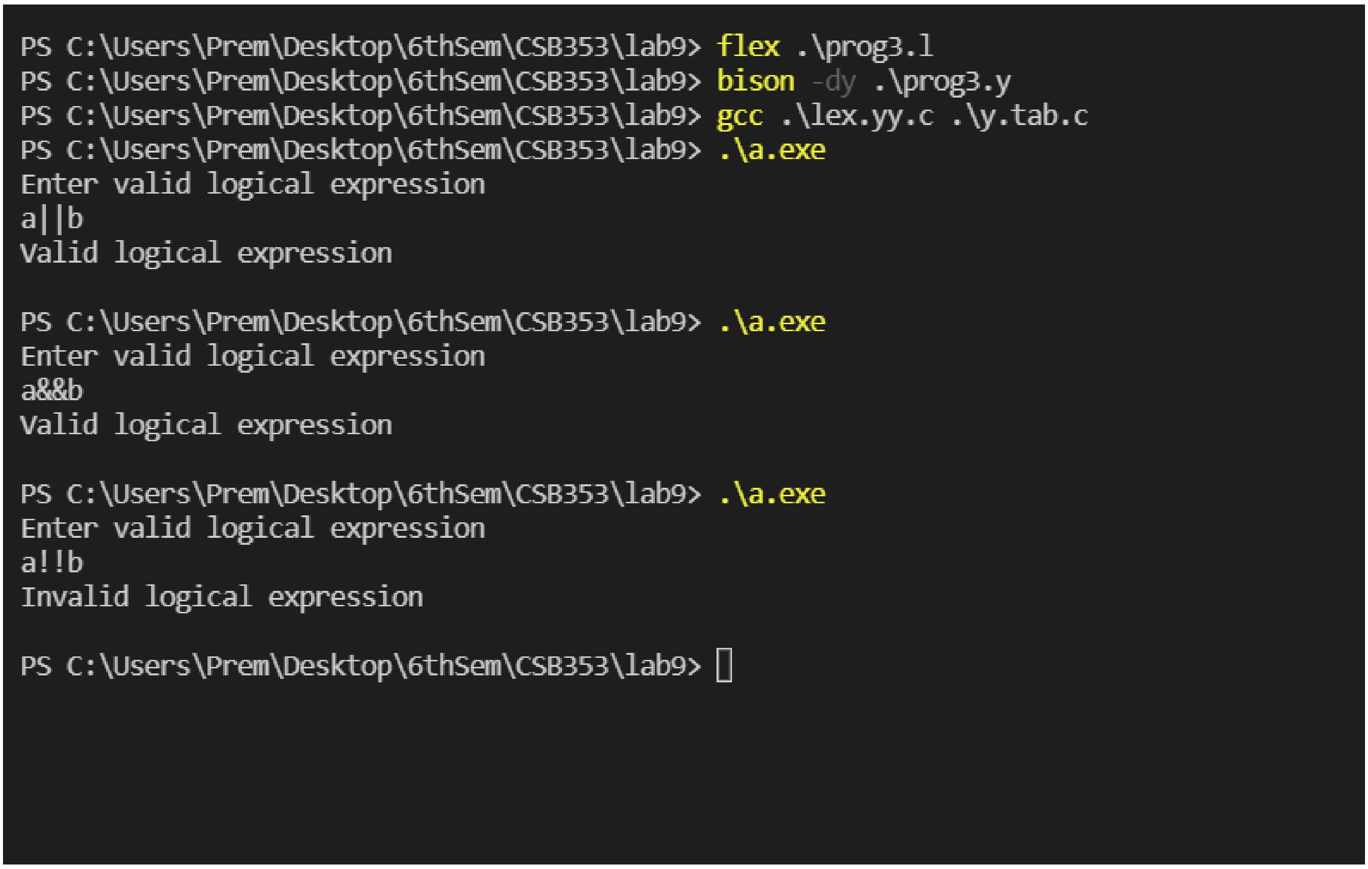
```
%{
#include "y.tab.h"
#include<stdio.h>
int yylval;
%}
%%
"&&"|"||" {return LOGICAL_OPERATOR;}
"(" {return OPENING_PARANTHESIS;}
")" {return CLOSING_PARANTHESIS;}
[a-zA-Z][a-zA-Z0-9]* {return ID;}
[0-9]+ {return NUM;}
[ \t] ;
\n {return 0;}
. {return yytext[0];}
%%
int yywrap(){
return 1;
}
```

- prog3.y (Yacc File)



```
%{
#include<stdio.h>
int yylex();
int yyerror(char* s);
%}

%token ID NUM OPENING_PARANTHESIS CLOSING_PARANTHESIS
%token LOGICAL_OPERATOR
%left LOGICAL_OPERATOR
%%
LogicalExpression : E {
printf("Valid logical expression\n\n");
return 0;
}
E : expr LOGICAL_OPERATOR expr | OPENING_PARANTHESIS E CLOSING_PARANTHESIS
expr : ID | NUM
%%
int main()
{
printf("Enter valid logical expression\n");
yyparse();
}
int yyerror(char* s)
{
printf("Invalid logical expression\n\n");
}
```

Output:



```
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> flex .\prog3.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> bison -dy .\prog3.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid logical expression
a||b
Valid logical expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid logical expression
a&&b
Valid logical expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> .\a.exe
Enter valid logical expression
a!!b
Invalid logical expression

PS C:\Users\Prem\Desktop\6thSem\CSB353\lab9> 
```