CSB 353: Compiler Design

LAB 12-13

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

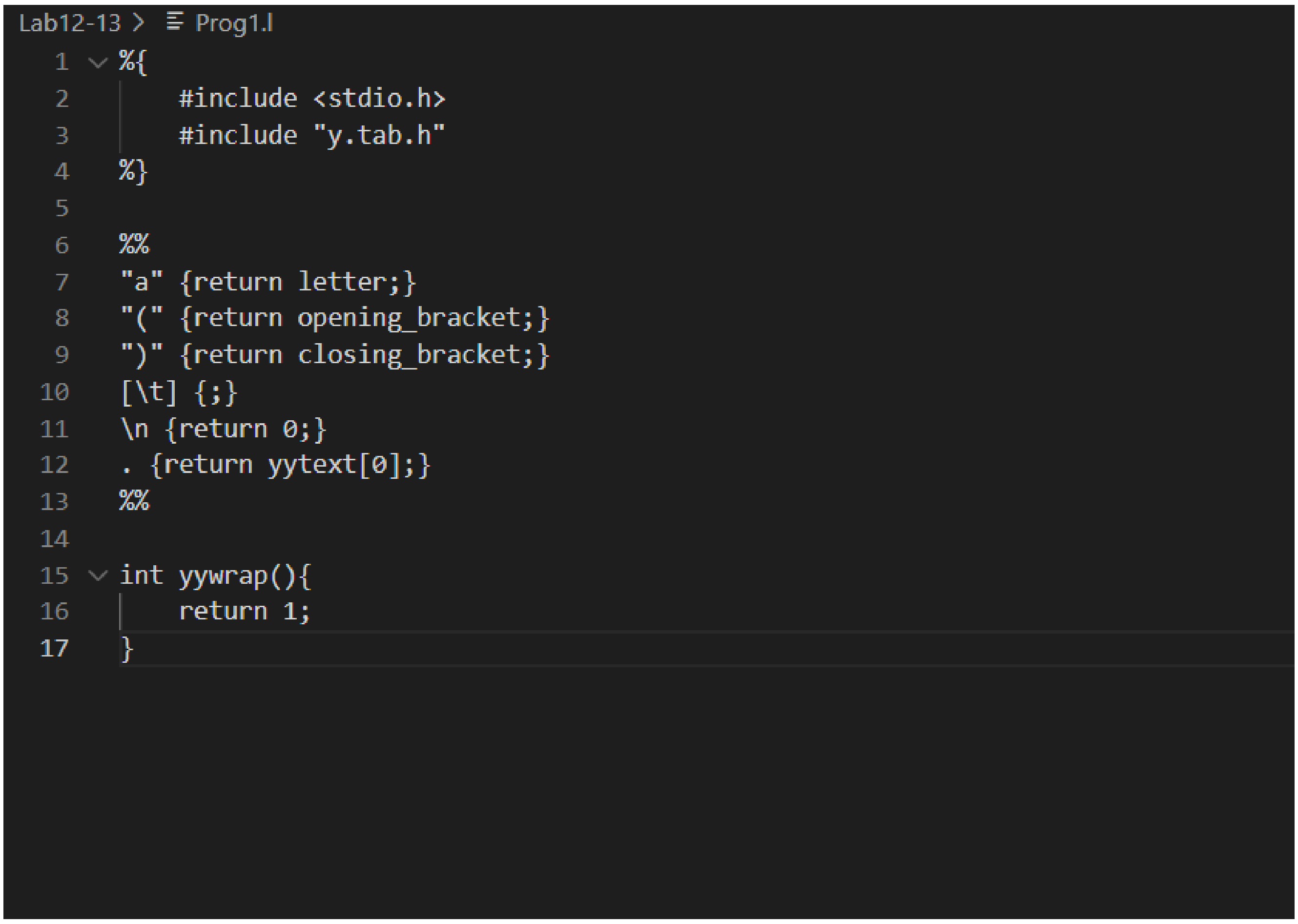
2019-2023

Ques 1. Write a yacc program to check if the parenthesis are balanced and count the number of matching parenthesis

P->(P) | a .

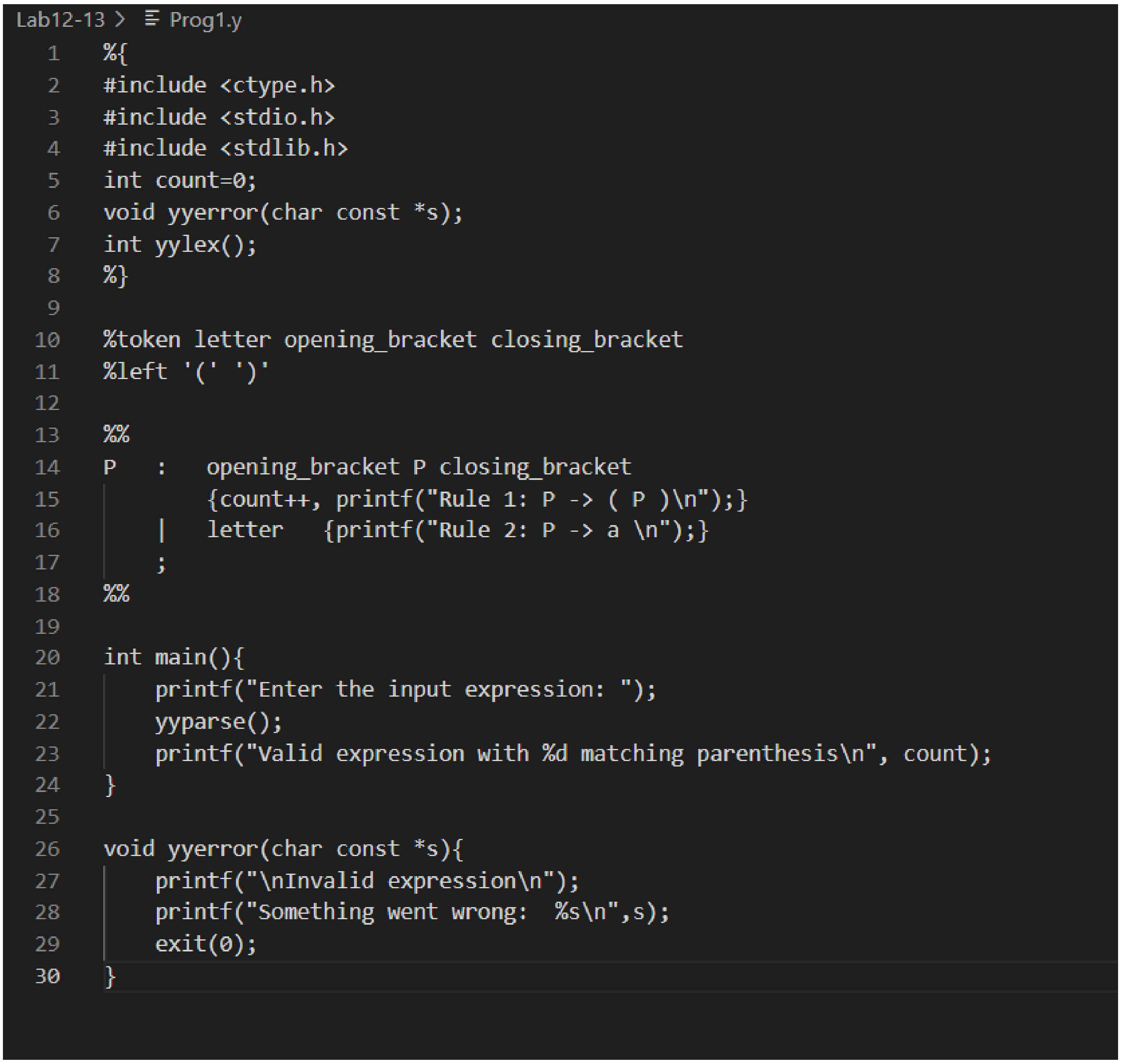
Code:

- prog1.l (Lex File)



```
%{
    #include <stdio.h>
    #include "y.tab.h"
%}

%%
"a" {return letter;}
"(" {return opening_bracket;}
")" {return closing_bracket;}
[\t] {;}
\n {return 0;}
. {return yytext[0];}
%%

int yywrap(){
    return 1;
}
```

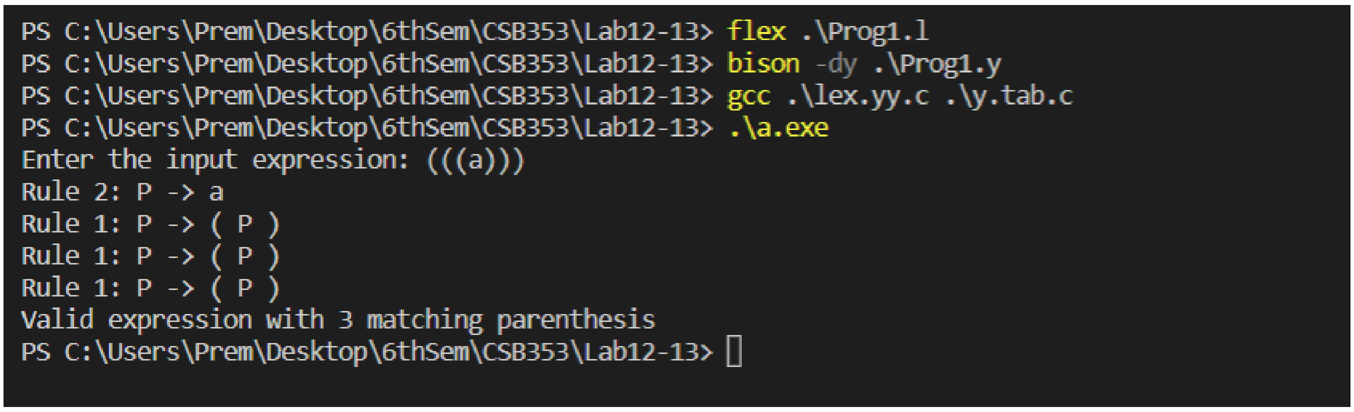
- prog1.y (Yacc File)



Lab12-13 > Prog1.y

```
%{
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
int count=0;
void yyerror(char const *s);
int yylex();
%}

%token letter opening_bracket closing_bracket
%left '(' ')'

%%
P   :   opening_bracket P closing_bracket
        {count++, printf("Rule 1: P -> ( P )\n");}
    |   letter   {printf("Rule 2: P -> a \n");}
    ;
%%

int main(){
    printf("Enter the input expression: ");
    yyparse();
    printf("Valid expression with %d matching parenthesis\n", count);
}

void yyerror(char const *s){
    printf("\nInvalid expression\n");
    printf("Something went wrong:  %s\n",s);
    exit(0);
}
```

- Output:



```
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> flex .\Prog1.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> bison -dy .\Prog1.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: (((a)))
Rule 2: P -> a
Rule 1: P -> ( P )
Rule 1: P -> ( P )
Rule 1: P -> ( P )
Valid expression with 3 matching parenthesis
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> 
```

Ques 2. Write a YACC program to recognize expressions involving binary numbers as operands and print the decimal value of the expression

> 1001

Decimal value: 9

>101 + 100

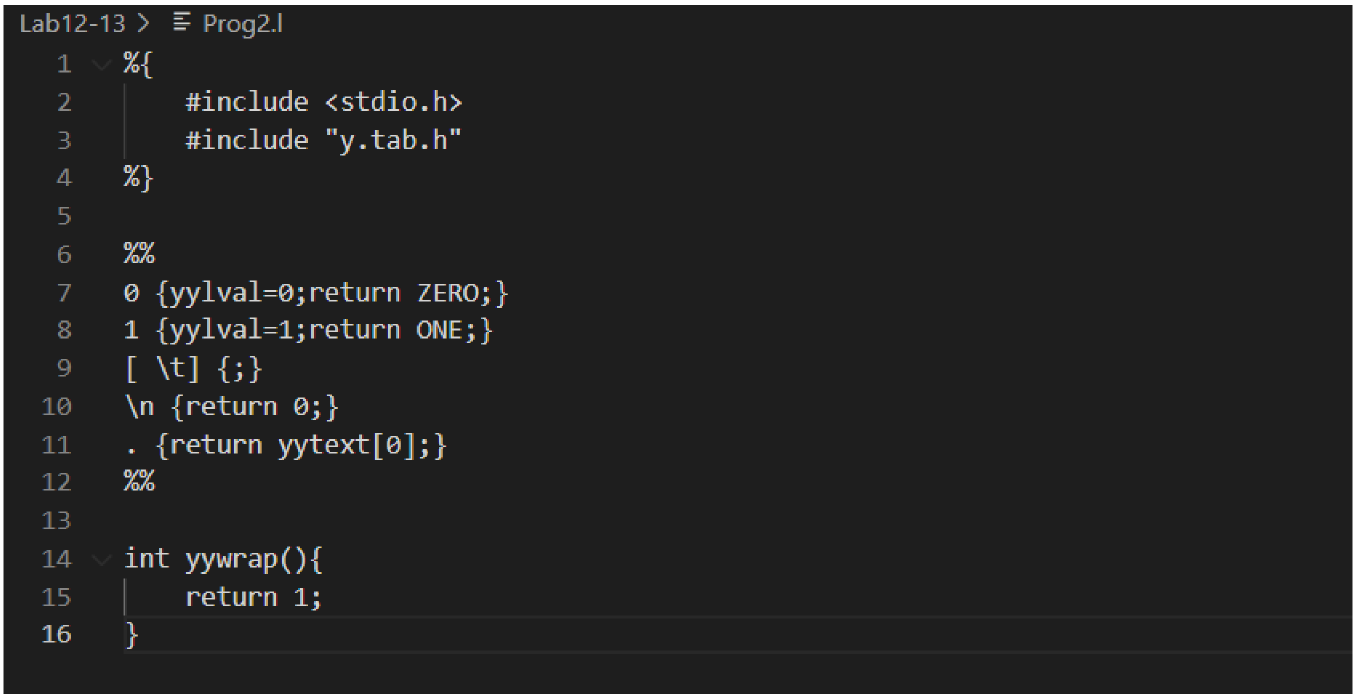
Decimal value: 9

> 201 + 101

Invalid input.

Code:

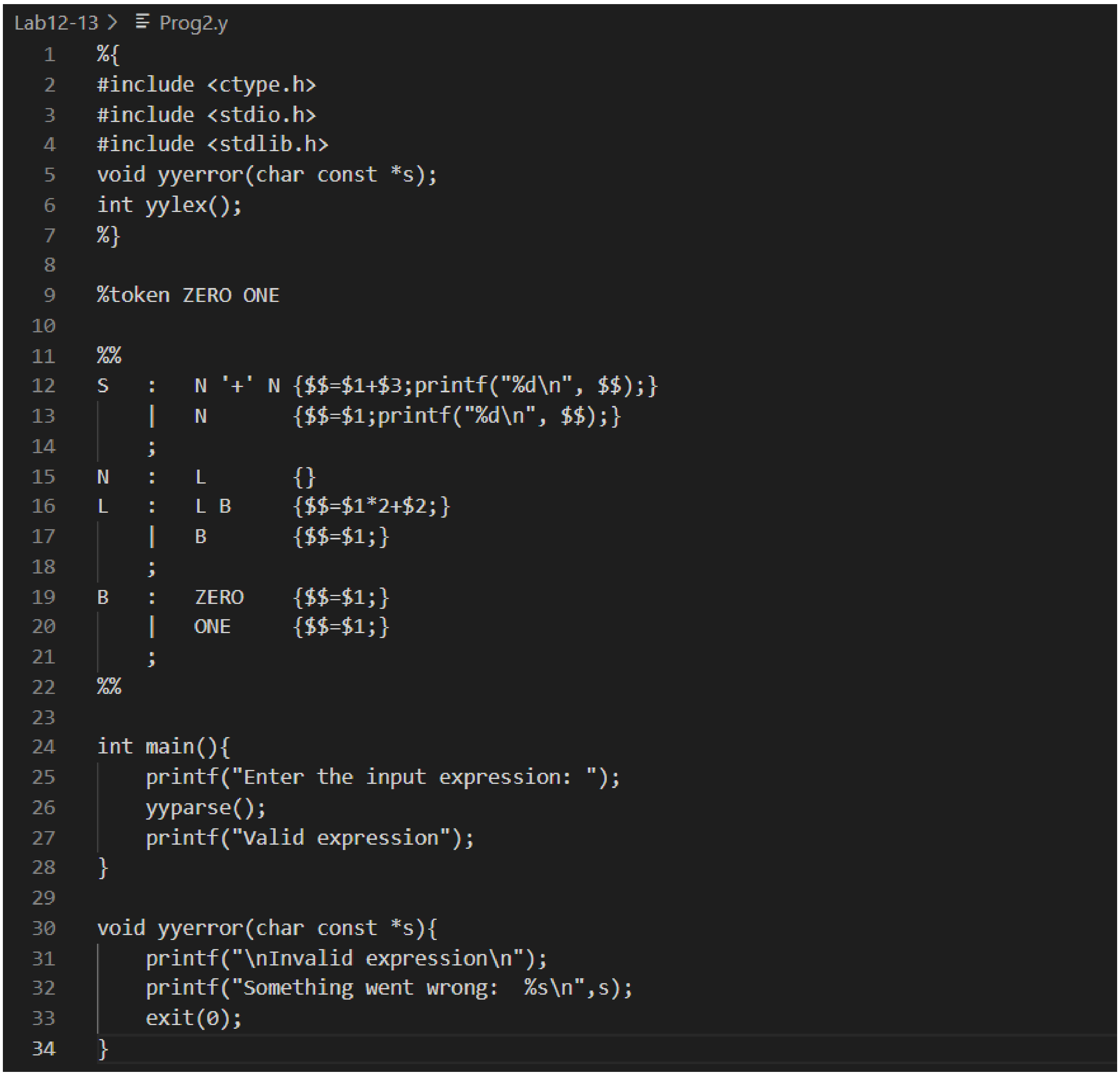
- prog2.l (Lex File)



Lab12-13 > Prog2.l

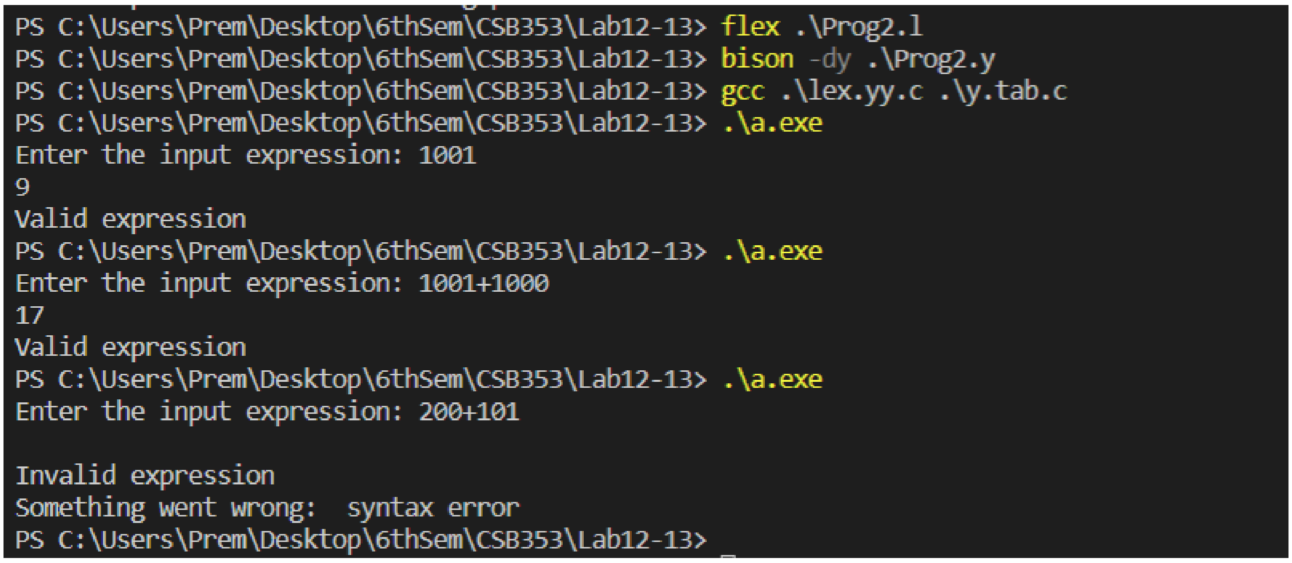
```
%{
    #include <stdio.h>
    #include "y.tab.h"
%}

%%
0 {yylval=0;return ZERO;}
1 {yylval=1;return ONE;}
[ \t] {;}
\n {return 0;}
. {return yytext[0];}
%%

int yywrap(){
    return 1;
}
```

- prog2.y (Yacc File)



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

%token ZERO ONE

%%
S   :   N '+' N {$$=$1+$3;printf("%d\n", $$);}
    |   N       {$$=$1;printf("%d\n", $$);}
    ;
N   :   L       {}
L   :   L B     {$$=$1*2+$2;}
    |   B       {$$=$1;}
    ;
B   :   ZERO    {$$=$1;}
    |   ONE     {$$=$1;}
    ;
%%

int main(){
    printf("Enter the input expression: ");
    yyparse();
    printf("Valid expression");
}

void yyerror(char const *s){
    printf("\nInvalid expression\n");
    printf("Something went wrong:  %s\n",s);
    exit(0);
}
```

- Output:



```
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> flex .\Prog2.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> bison -dy .\Prog2.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: 1001
9
Valid expression
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: 1001+1000
17
Valid expression
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: 200+101

Invalid expression
Something went wrong:  syntax error
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> _
```

Ques 3. Append semantic actions to the grammar of simple desk calculator that performs simple operations on integer expressions with the grammar:

exp -> exp addop term | term

addop -> + | -

term -> term mulop factor | factor

mulop -> *

factor -> (exp) | number

number -> number digit | digit

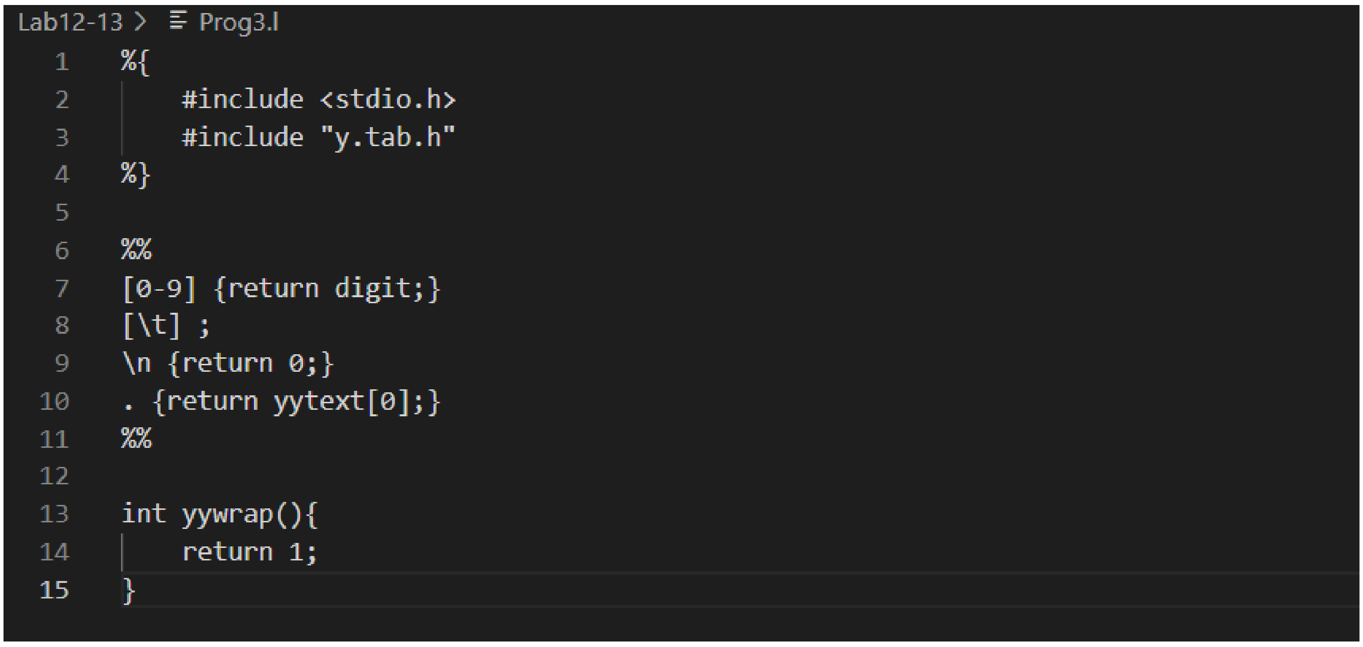
digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

You are required to write YACC specifications for this grammar so that the parser evaluates any arithmetic expressions and the output shows each semantic action as it is applied in the parsing process. Show your parsing sequence for the input string:

(2+(3*4)).

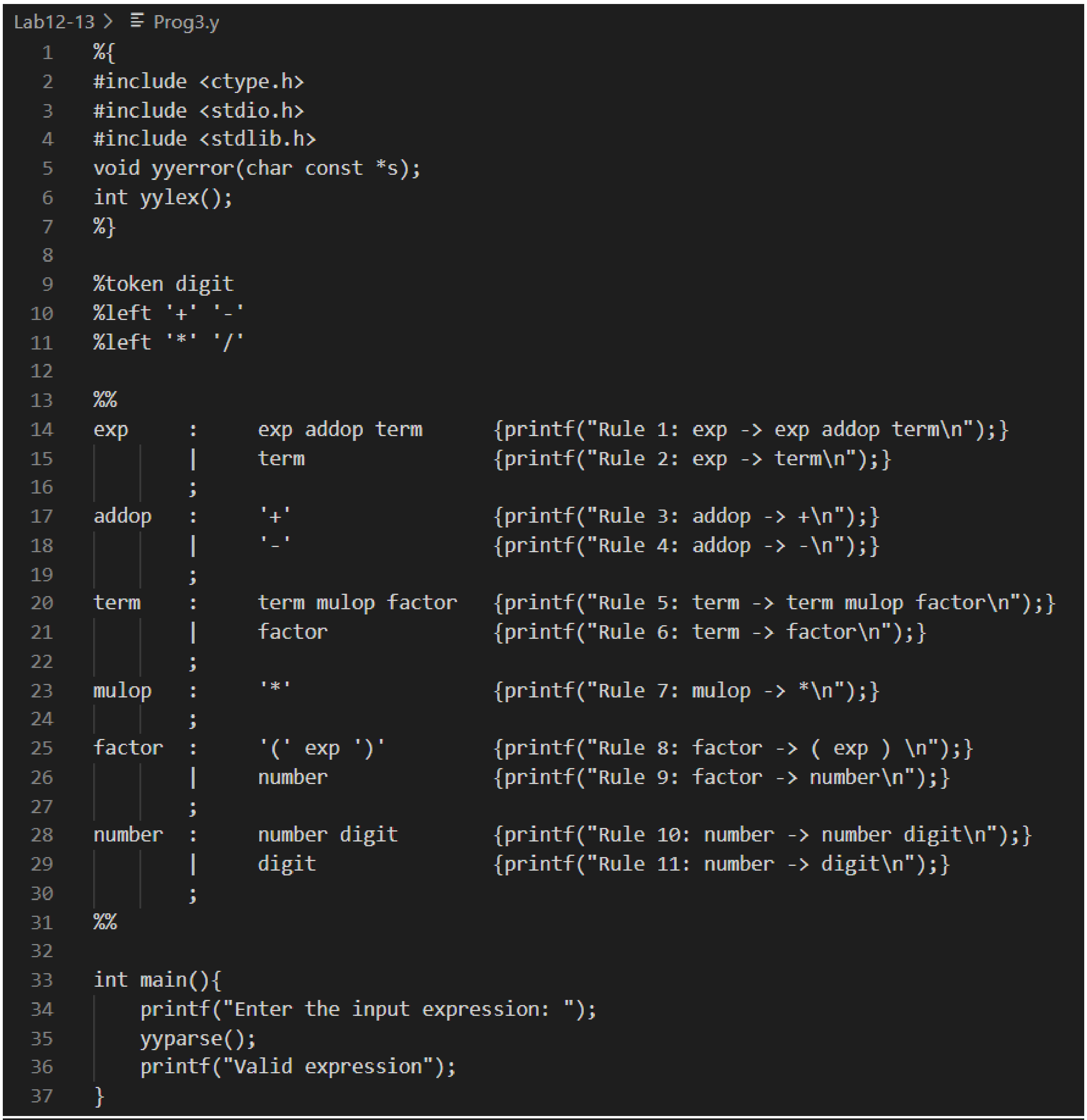
Code:

- prog3.l (Lex File)

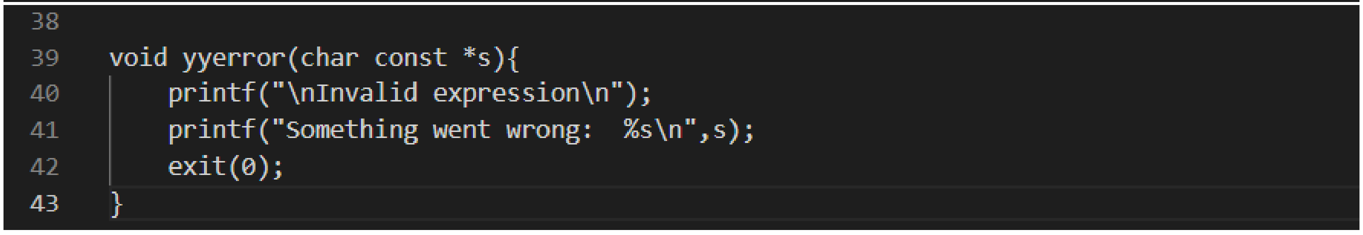

Lab12-13 > Prog3.l

```
%{
    #include <stdio.h>
    #include "y.tab.h"
%}

%%
[0-9] {return digit;}
[\t] ;
\n {return 0;}
. {return yytext[0];}
%%

int yywrap(){
    return 1;
}
```

- prog3.y (Yacc File)

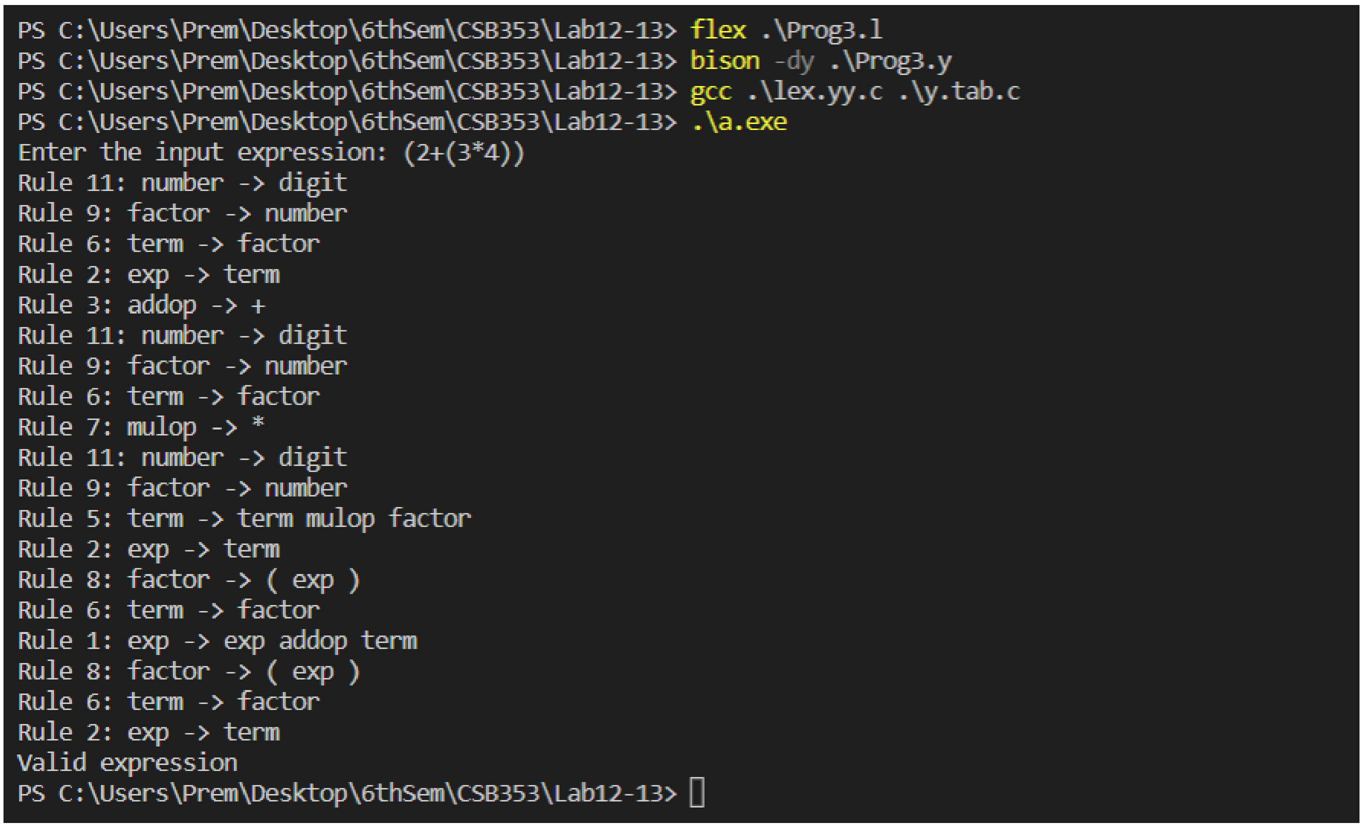


```
%{
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
void yyerror(char const *s);
int yylex();
%}

%token digit
%left '+' '-'
%left '*' '/'

%%
exp     :    exp addop term      {printf("Rule 1: exp -> exp addop term\n");}
        |    term                {printf("Rule 2: exp -> term\n");}
        ;
addop   :    '+'                 {printf("Rule 3: addop -> +\n");}
        |    '-'                 {printf("Rule 4: addop -> -\n");}
        ;
term    :    term mulop factor   {printf("Rule 5: term -> term mulop factor\n");}
        |    factor              {printf("Rule 6: term -> factor\n");}
        ;
mulop   :    '*'                 {printf("Rule 7: mulop -> *\n");}
        ;
factor  :    '(' exp ')'         {printf("Rule 8: factor -> ( exp ) \n");}
        |    number              {printf("Rule 9: factor -> number\n");}
        ;
number  :    number digit        {printf("Rule 10: number -> number digit\n");}
        |    digit               {printf("Rule 11: number -> digit\n");}
        ;
%%

int main(){
    printf("Enter the input expression: ");
    yyparse();
    printf("Valid expression");
}
```



```

void yyerror(char const *s){
    printf("\nInvalid expression\n");
    printf("Something went wrong:  %s\n",s);
    exit(0);
}
```

- Output:



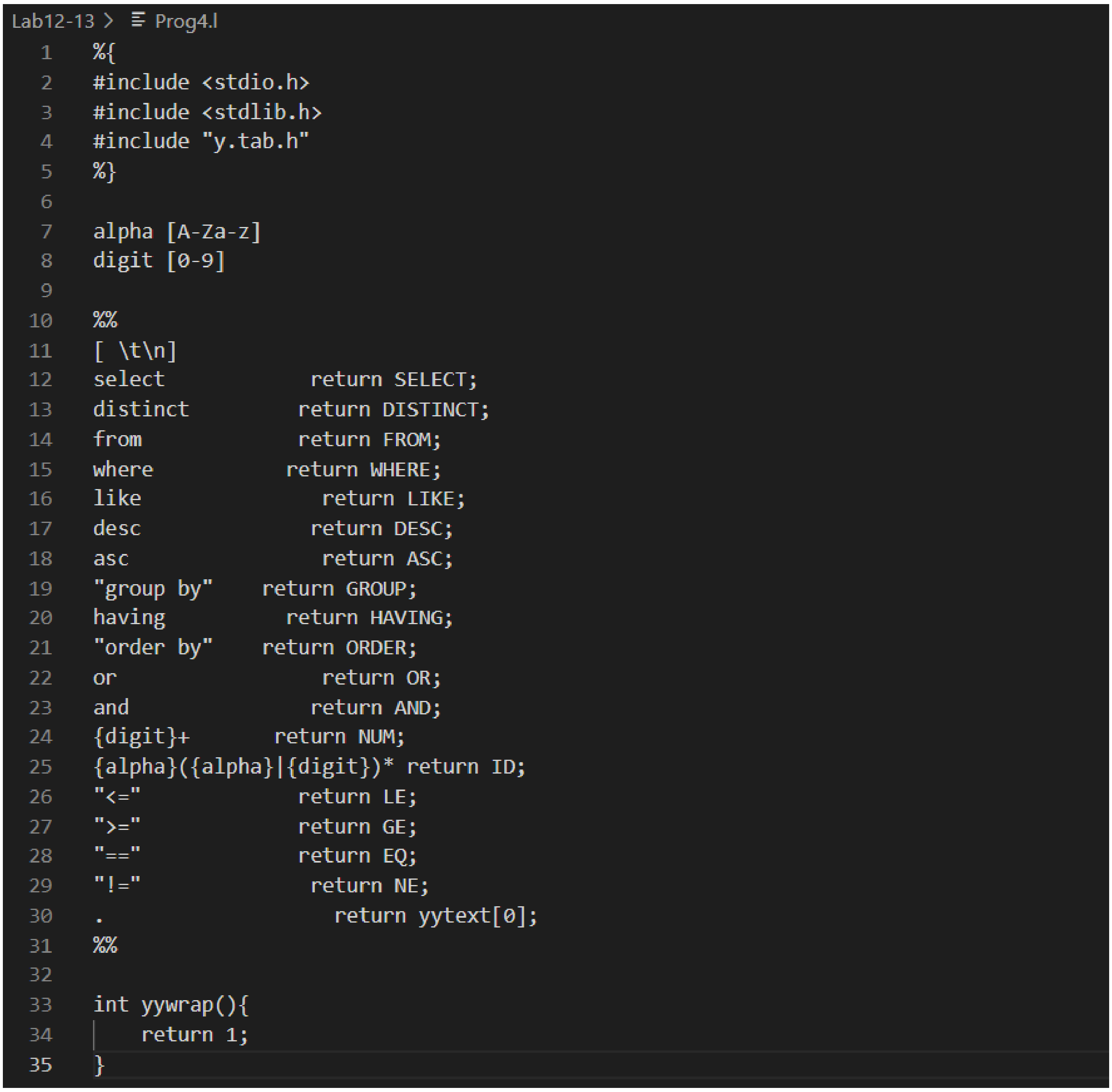
```
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> flex .\Prog3.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> bison -dy .\Prog3.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: (2+(3*4))
Rule 11: number -> digit
Rule 9: factor -> number
Rule 6: term -> factor
Rule 2: exp -> term
Rule 3: addop -> +
Rule 11: number -> digit
Rule 9: factor -> number
Rule 6: term -> factor
Rule 7: mulop -> *
Rule 11: number -> digit
Rule 9: factor -> number
Rule 5: term -> term mulop factor
Rule 2: exp -> term
Rule 8: factor -> ( exp )
Rule 6: term -> factor
Rule 1: exp -> exp addop term
Rule 8: factor -> ( exp )
Rule 6: term -> factor
Rule 2: exp -> term
Valid expression
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13>
```

Ques 4. Generate a parser for sql select statement (select statement may contain group by, having clause)and do the type checking wherever required. Count the number of attributes selected and number of attributes used in conditions.

For eg.: select name,address from emp where age>20 group by name having age<40 order by name desc;

Code:

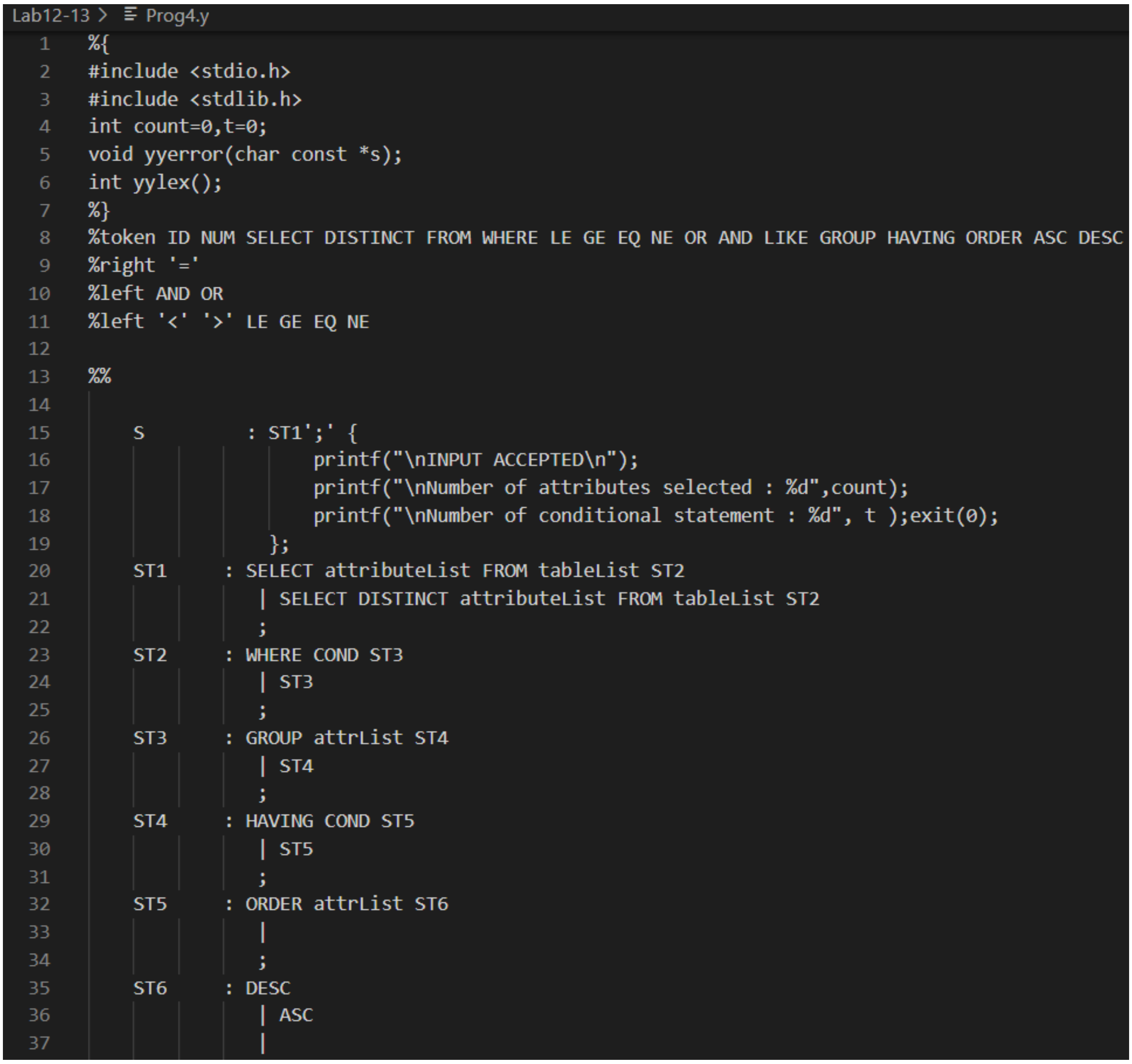
- prog4.l (Lex File)



Lab12-13 > Prog4.l

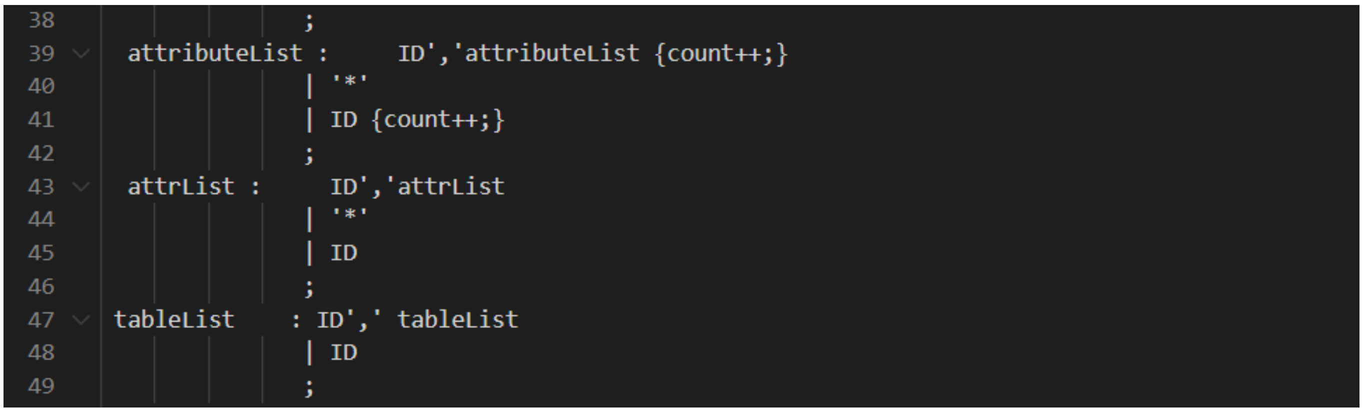
```
%{
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
%}

alpha [A-Za-z]
digit [0-9]

%%
[ \t\n]
select             return SELECT;
distinct          return DISTINCT;
from              return FROM;
where            return WHERE;
like                return LIKE;
desc               return DESC;
asc                 return ASC;
"group by"    return GROUP;
having           return HAVING;
"order by"    return ORDER;
or                  return OR;
and                return AND;
{digit}+       return NUM;
{alpha}({alpha}|{digit})* return ID;
"<="              return LE;
">="              return GE;
"=="              return EQ;
"!="               return NE;
.                    return yytext[0];
%%

int yywrap(){
    return 1;
}
```

- prog4.y (Yacc File)

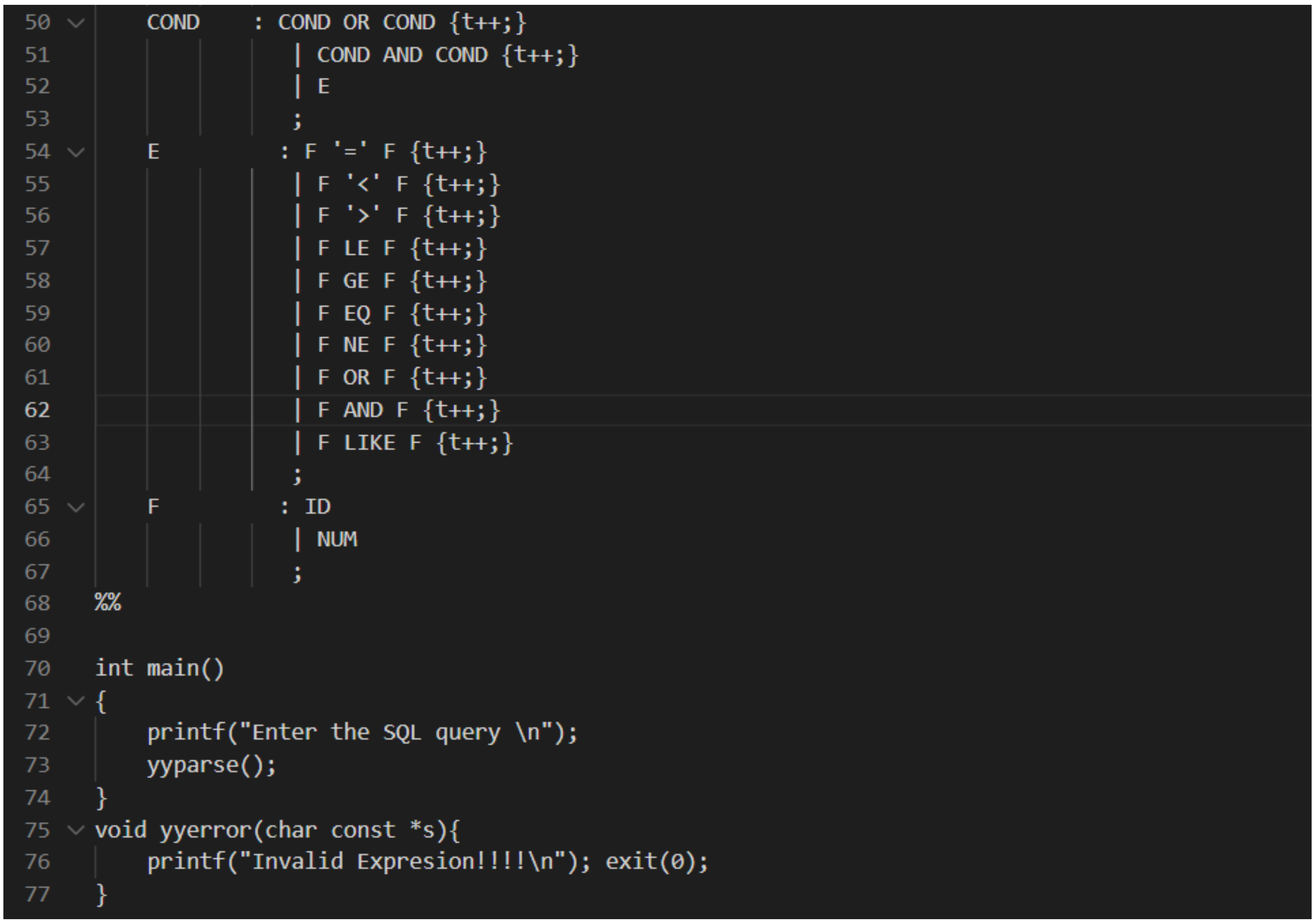


Lab12-13 > Prog4.y

```
%{
#include <stdio.h>
#include <stdlib.h>
int count=0,t=0;
void yyerror(char const *s);
int yylex();
%}
%token ID NUM SELECT DISTINCT FROM WHERE LE GE EQ NE OR AND LIKE GROUP HAVING ORDER ASC DESC
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE

%%

    S         : ST1';' {
                    printf("\nINPUT ACCEPTED\n");
                    printf("\nNumber of attributes selected : %d",count);
                    printf("\nNumber of conditional statement : %d", t );exit(0);
                };
    ST1     : SELECT attributeList FROM tableList ST2
              | SELECT DISTINCT attributeList FROM tableList ST2
              ;
    ST2     : WHERE COND ST3
              | ST3
              ;
    ST3     : GROUP attrList ST4
              | ST4
              ;
    ST4     : HAVING COND ST5
              | ST5
              ;
    ST5     : ORDER attrList ST6
              |
              ;
    ST6     : DESC
              | ASC
              |
```

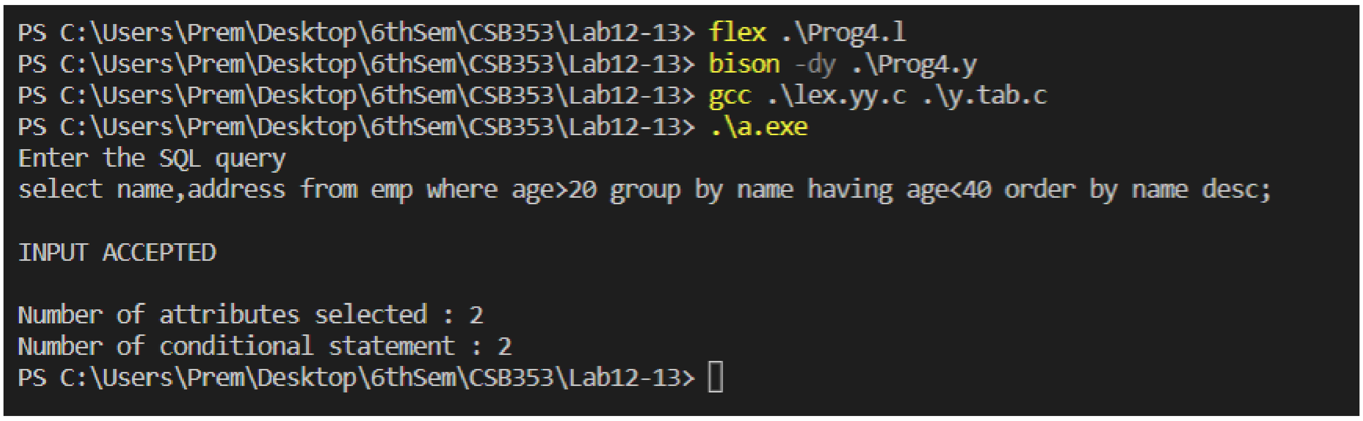


```
              ;
  attributeList :     ID','attributeList {count++;}
              | '*'
              | ID {count++;}
              ;
  attrList :     ID','attrList
              | '*'
              | ID
              ;
 tableList    : ID',' tableList
              | ID
              ;
```



```
    COND    : COND OR COND {t++;}
              | COND AND COND {t++;}
              | E
              ;
    E         : F '=' F {t++;}
              | F '<' F {t++;}
              | F '>' F {t++;}
              | F LE F {t++;}
              | F GE F {t++;}
              | F EQ F {t++;}
              | F NE F {t++;}
              | F OR F {t++;}
              | F AND F {t++;}
              | F LIKE F {t++;}
              ;
    F         : ID
              | NUM
              ;
%%

int main()
{
    printf("Enter the SQL query \n");
    yyparse();
}
void yyerror(char const *s){
    printf("Invalid Expresion!!!!\n"); exit(0);
}
```

- Output:



```
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> flex .\Prog4.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> bison -dy .\Prog4.y
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the SQL query
select name,address from emp where age>20 group by name having age<40 order by name desc;

INPUT ACCEPTED

Number of attributes selected : 2
Number of conditional statement : 2
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13>
```

Ques 5. Write a YACC program to evaluate the following expressions. Specify the grammar clearly.

Functions Meaning

(+ x1 x2 x3 ….. xn) Calculate Sum of x1, x2 upto xn

(* x1 x2 x3 ….. xn) Calculate Product of x1, x2 upto xn

(max x1 x2 x3 ….. xn) Calculate Maximum of x1, x2 upto xn

(min x1 x2 x3 ….. xn) Calculate Minimum of x1, x2 upto xn

Sample Input:

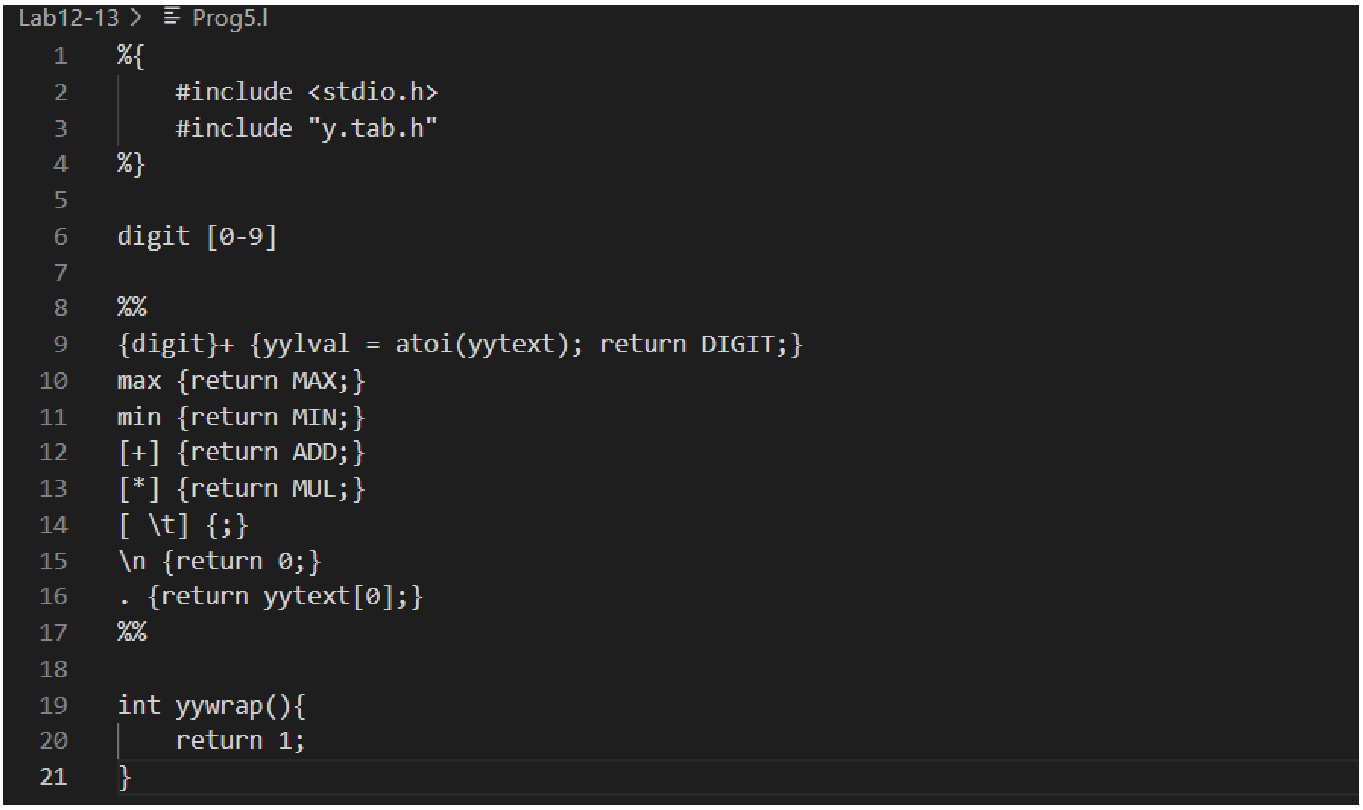
(+ 6 12 18)

Sample Output:

36.

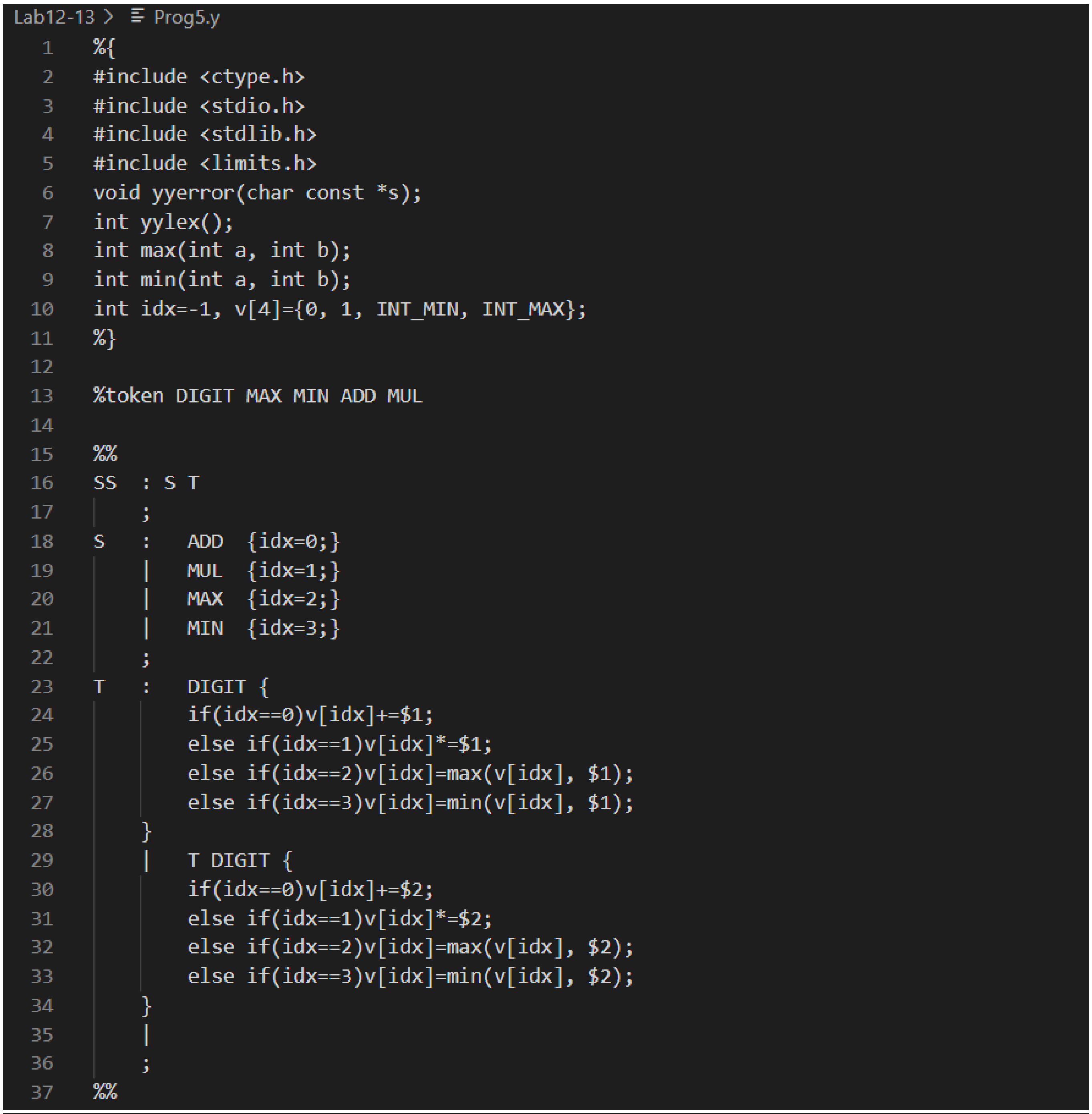
Code:

- prog5.l (Lex File)



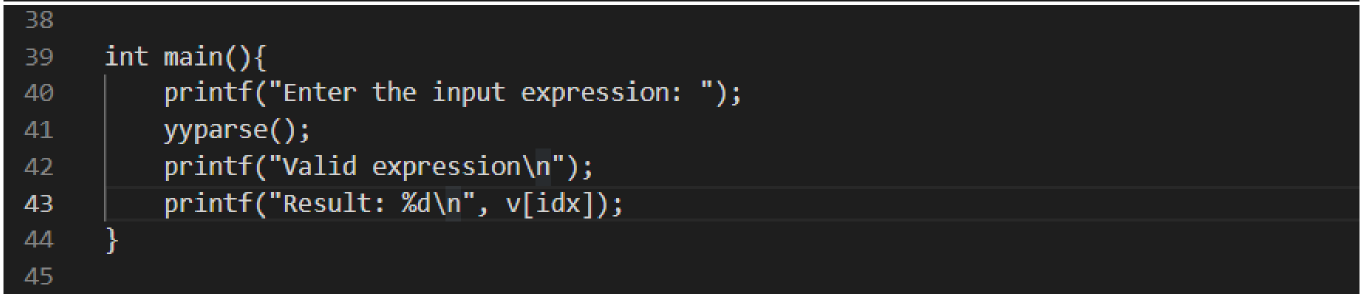
```
%{
    #include <stdio.h>
    #include "y.tab.h"
%}

digit [0-9]

%%
{digit}+ {yylval = atoi(yytext); return DIGIT;}
max {return MAX;}
min {return MIN;}
[+] {return ADD;}
[*] {return MUL;}
[ \t] {;}
\n {return 0;}
. {return yytext[0];}
%%

int yywrap(){
    return 1;
}
```

- prog5.y (Yacc File)

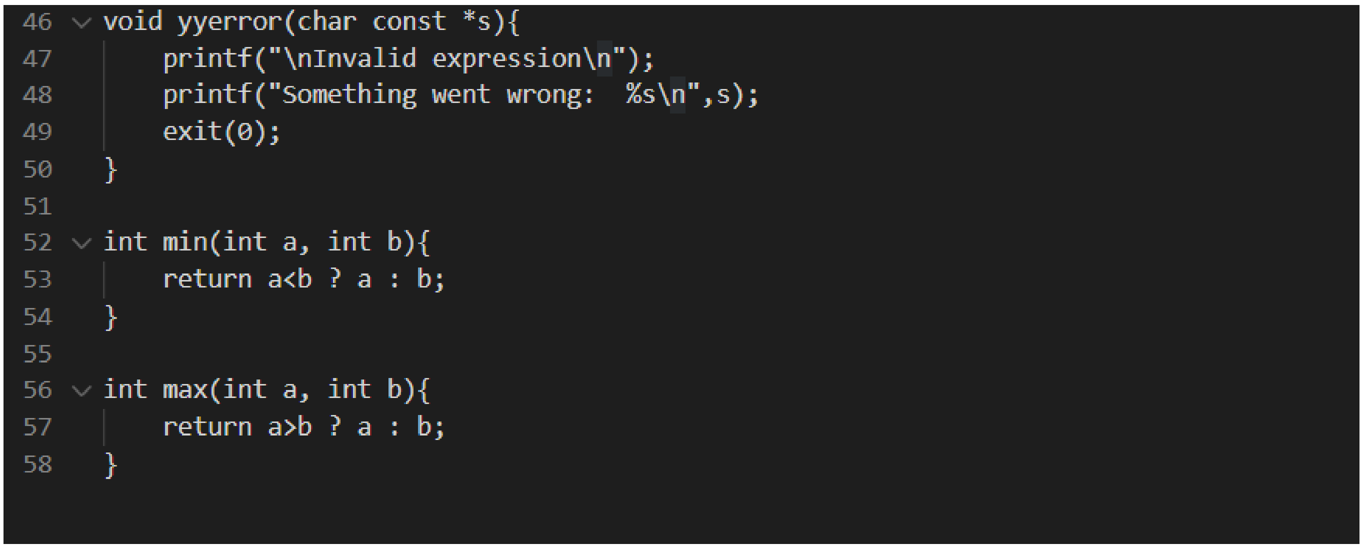


Lab12-13 > Prog5.y

```
%{
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
void yyerror(char const *s);
int yylex();
int max(int a, int b);
int min(int a, int b);
int idx=-1, v[4]={0, 1, INT_MIN, INT_MAX};
%}

%token DIGIT MAX MIN ADD MUL

%%
SS  : S T
    ;
S   :   ADD  {idx=0;}
    |   MUL  {idx=1;}
    |   MAX  {idx=2;}
    |   MIN  {idx=3;}
    ;
T   :   DIGIT {
        if(idx==0)v[idx]+=$1;
        else if(idx==1)v[idx]*=$1;
        else if(idx==2)v[idx]=max(v[idx], $1);
        else if(idx==3)v[idx]=min(v[idx], $1);
    }
    |   T DIGIT {
        if(idx==0)v[idx]+=$2;
        else if(idx==1)v[idx]*=$2;
        else if(idx==2)v[idx]=max(v[idx], $2);
        else if(idx==3)v[idx]=min(v[idx], $2);
    }
    |
    ;
%%
```

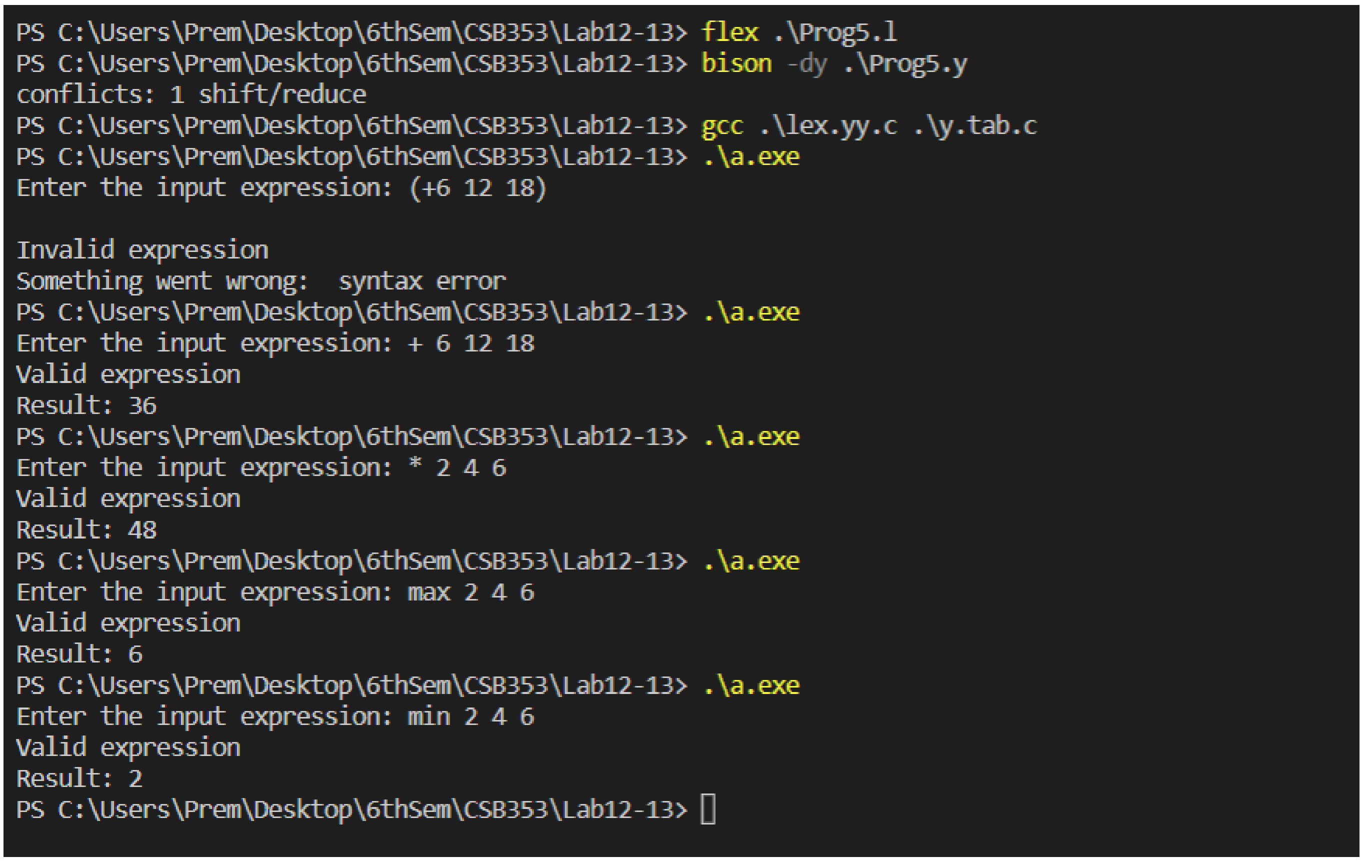


```

int main(){
    printf("Enter the input expression: ");
    yyparse();
    printf("Valid expression\n");
    printf("Result: %d\n", v[idx]);
}

```



```c
void yyerror(char const *s){
    printf("\nInvalid expression\n");
    printf("Something went wrong:  %s\n",s);
    exit(0);
}

int min(int a, int b){
    return a<b ? a : b;
}

int max(int a, int b){
    return a>b ? a : b;
}
```

- Output:



```
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> flex .\Prog5.l
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> bison -dy .\Prog5.y
conflicts: 1 shift/reduce
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> gcc .\lex.yy.c .\y.tab.c
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: (+6 12 18)

Invalid expression
Something went wrong:  syntax error
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: + 6 12 18
Valid expression
Result: 36
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: * 2 4 6
Valid expression
Result: 48
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: max 2 4 6
Valid expression
Result: 6
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> .\a.exe
Enter the input expression: min 2 4 6
Valid expression
Result: 2
PS C:\Users\Prem\Desktop\6thSem\CSB353\Lab12-13> 
```