Compiler
Design

Practical 2: Lexical Analysis
+ Syntax Analysis

Name : Himani Dighorikar
Roll No. 03

Aim :

Instructor Led Practical:

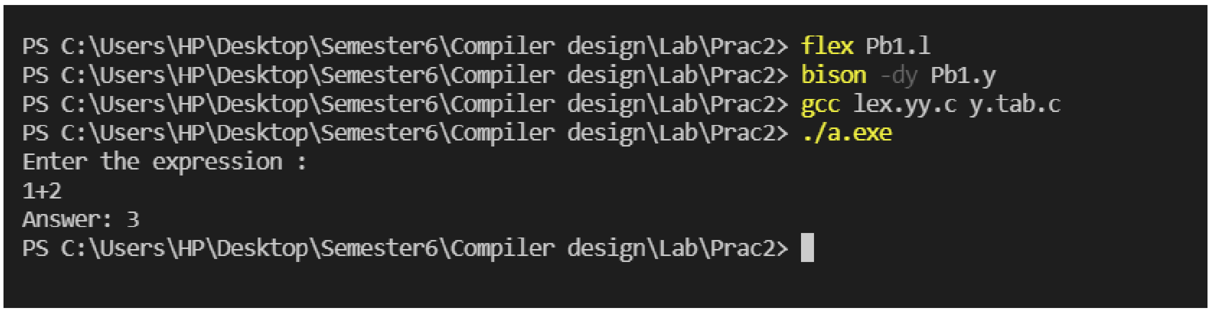
I1: Write YACC specification to check syntax of a simple expression involving operators +, -, * and / and evaluate the expression.

Code :

```
%{
#include "y.tab.h"
%}

%%
[0-9]+  {yylval=atoi(yytext);
        return NUMBER;
        }
[a-zA-Z]  {
    return ID;
}
\n {return NL;}
. {return yytext[0];}
%%
```

```
%{

#include<stdio.h>
#include<stdlib.h>
int answer=0;

%}

// bison -dy 1a.y
// gcc lex.yy.c y.tab.c
// -dy for creating headers

%token NUMBER ID NL
%left '+' '-'
```

```
%left '*' '/'
%%

stmt :
exp NL {printf("Answer: %d \n",$1); exit(0);}
|
exp1 NL {printf("valid expression \nBut, calculation can't be performed on
variables \n");
            exit(0);}

exp : exp '+' exp {$$=$1+$3;}
| exp '-' exp {$$=$1-$3;}
| exp '*' exp  {$$=$1*$3;}
| exp '/' exp {$$=$1/$3;}
| '(' exp ')'
| NUMBER ;

exp1 :  exp1 '+' exp1
| exp1 '-' exp1
| exp1 '*' exp1
| exp1 '/' exp1
| '(' exp1 ')'
| ID ;
%%

int yyerror(char *msg){
    printf("Invalid Expression\n");
    exit(0);
}

int main(){
printf("Enter the expression : \n");
    return yyparse();
}
int yywrap(){
    return 1;
}
```



```
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> flex Pb1.l
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> bison -dy Pb1.y
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> gcc lex.yy.c y.tab.c
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the expression :
1+2
Answer: 3
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2>
```

Evaluation

E1: Write YACC specification to check syntax of a simple expression involving operators +, -, * and /. Also convert the arithmetic expression to postfix.

File E1.l

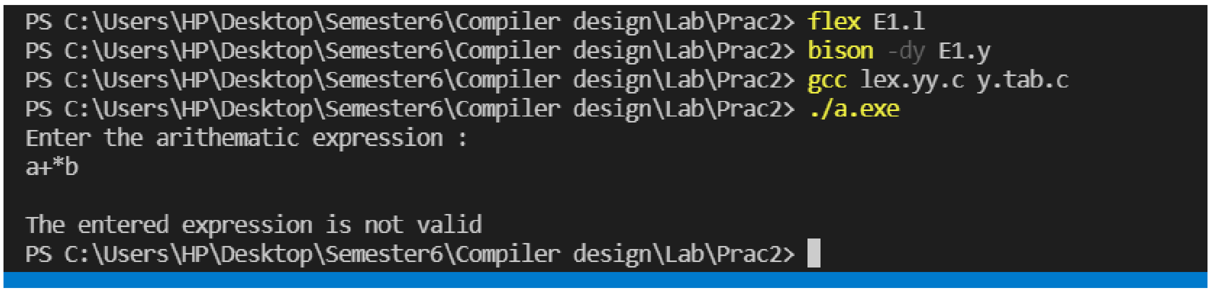
```
%{
#include "y.tab.h"
%}

%%
[0-9]  {        yylval=atoi(yytext);
        return NUMBER;
        }
[a-zA-Z] {yylval= atoi(yytext);
                return ID;}
\n {return NL;}
. {return yytext[0];}
%%
```

File E1.y

```
%{
#include<stdio.h>
#include<stdlib.h>
int answer=0;
%}

%token ID NUMBER NL
%left '+' '-'
%left '*' '/'

%%
stmt :
exp1 NL {
    printf("\nThe entered expression is valid");
            exit(0);
}
exp1 :  exp1 '+' exp1  { printf(" + "); }
| exp1 '-' exp1 { printf(" - "); }
| exp1 '*' exp1 { printf(" * "); }
| exp1 '/' exp1 { printf(" / "); }
| '(' exp1 ')'
| NUMBER { printf("%d",yylval);}
%%
```

```
int yyerror(char *msg){
    printf("\nThe entered expression is not valid");
    exit(0);
}

int main(){
printf("Enter the arithematic expression : \n");
return yyparse();

}
int yywrap(){

    return 1;
}
```

Output



```
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> flex E1.l
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> bison -dy E1.y
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> gcc lex.yy.c y.tab.c
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the arithematic expression :
a+*b

The entered expression is not valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2>
```

E2. Write a YACC specification to accept strings that starts and ends with 0 or 1

Code :

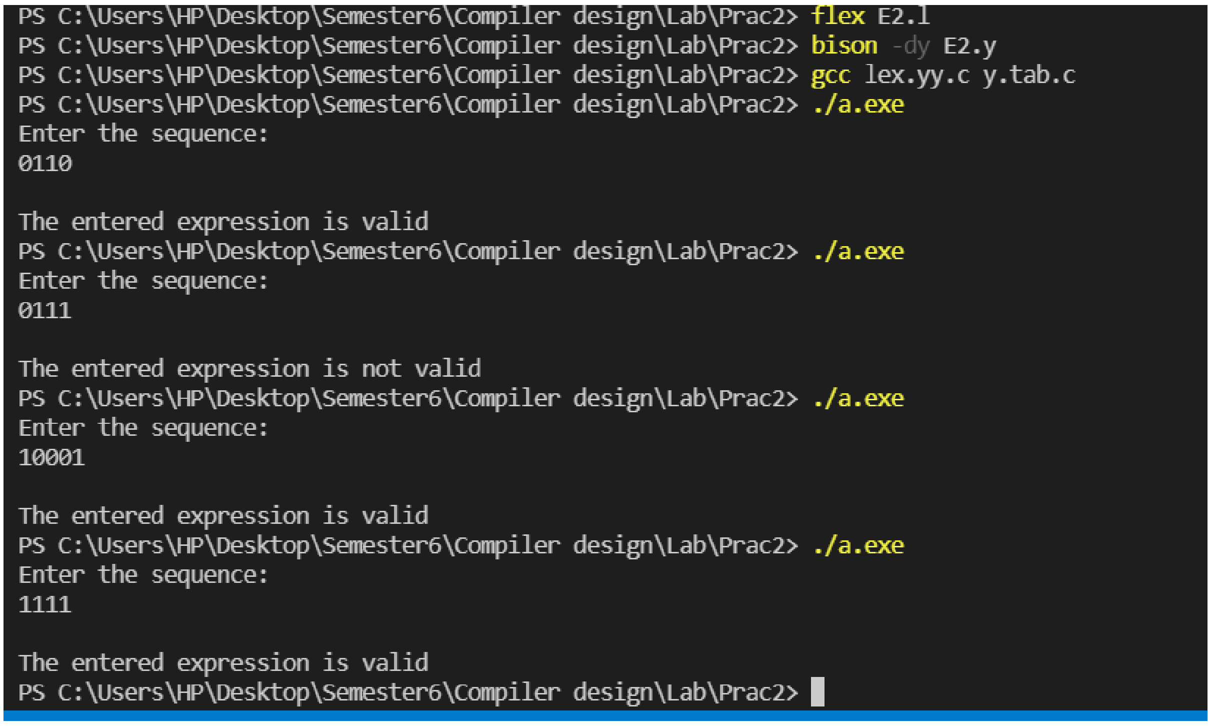
File E2.l

```
%{
#include "y.tab.h"
%}

%%
0 {return ZERO;}
1 {return ONE;}
\n {return NL;}
. {return yytext[0];}
%%
```

File E2.y

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

%}

%token ZERO ONE NL
%left '+' '-'
%left '*' '/'

%%
stmt :
eval NL {
    printf("\nThe entered expression is valid");
            exit(0);
}
eval : ZERO z
| ONE o
;

z : any z
|  ZERO
;

o : any o
|  ONE
;

any : ZERO
| ONE
;

%%

int yyerror(char *msg){
    printf("\nThe entered expression is not valid");
    exit(0);
}

int main(){
printf("Enter the sequence: \n");
return yyparse();

}
int yywrap(){

    return 1;
}
```

Output



```
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> flex E2.l
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> bison -dy E2.y
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> gcc lex.yy.c y.tab.c
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the sequence:
0110

The entered expression is valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the sequence:
0111

The entered expression is not valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the sequence:
10001

The entered expression is valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the sequence:
1111

The entered expression is valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2>
```

E3. Write a YACC specification to validate the string having general form as below. Construct a proper grammar for the same and also write the corresponding LEX.:

(a) Any alphabet(s) @ any alphabet + any digit – any digit.

(b) Date

(c) Expression of the form a=b*c

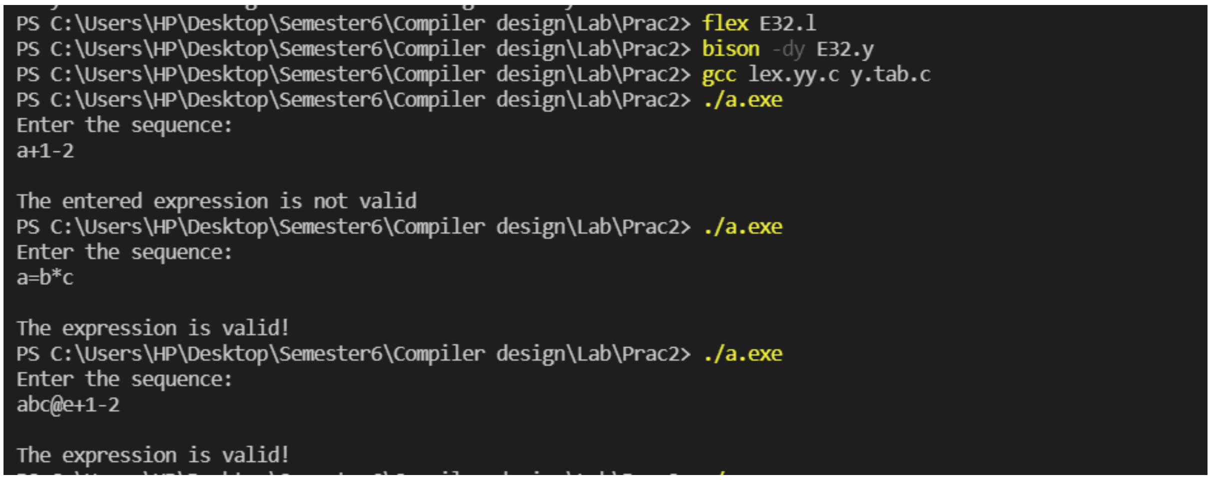
File E32.l

```
%{
#include "y.tab.h"
%}

%%

[a-zA-Z]    {return ID;}
[0-9]       {return NUMBER;}
\n      {return NL;}
.       {return yytext[0];}
```

```
%%
```

File E32.y

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

%token ID NUMBER NL
%left '+' '-'
%left '*' '/'

%%
stmt : CHECK NL {printf("\nThe expression is valid!\n"); exit(0);}
| CHECK_EXP NL {printf("\nThe expression is valid!\n"); exit(0);}
// | CHECK_DATE NL {printf("The date is valid!\n"); exit(0);}
;



CHECK_EXP : ID EQUATE;
EQUATE : '=' EXPR;
EXPR : ID MUL;
MUL : '*' ID;

CHECK : ID FOLLOW
;
FOLLOW : ID FOLLOW
| SPCHR
;
SPCHR : '@' ALPHA
;
ALPHA : ID ADD;
ADD : '+' SUB;
SUB : NUMBER LAST;
LAST : '-' NUMBER;

%%

int yyerror(char *msg){
    printf("\nThe entered expression is not valid");
    exit(0);
}

int main(){
printf("Enter the sequence: \n");
return yyparse();

}
int yywrap(){

    return 1;
}
```



```
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> flex E32.l
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> bison -dy E32.y
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> gcc lex.yy.c y.tab.c
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the sequence:
a+1-2

The entered expression is not valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the sequence:
a=b*c

The expression is valid!
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the sequence:
abc@e+1-2

The expression is valid!
```

E4. To validate syntax of following programing language construct:

Batch B1: do while loop

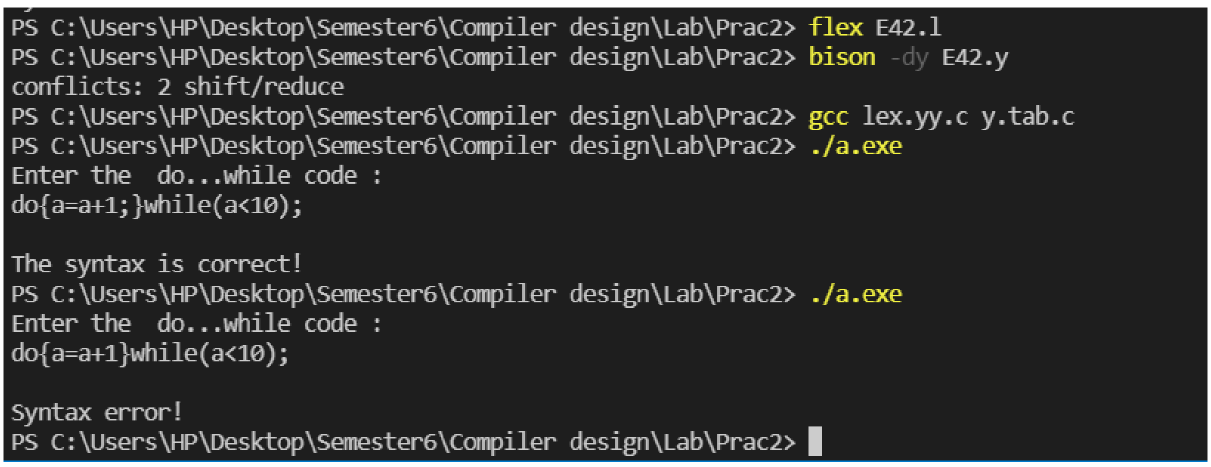
File E42.l

```
%{
#include "y.tab.h"
%}

%%
[ \t\n]
do     return DO;
while     return WHILE;
[0-9]+    return NUMBER;
[a-z]([a-z]|[0-9])*     return ID;
"<="     return LE;
">="     return GE;
"=="     return EQ;
"!="     return NE;
"||"     return OR;
"&&"     return AND;
.    return yytext[0];
%%
```

File E42.y

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUMBER DO WHILE LE GE EQ NE OR AND
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/'
%right UMINUS
%left '!'

%%

loop : block {printf("\nThe syntax is correct!\n");exit(0);};

block : DO '{' stmt '}' WHILE'(' E2 ')'';';

stmt :stmt stmt
| E ';'
;
E : E LE E
| E GE E
| E EQ E
| E NE E
| E OR E
| E AND E
|ID'='E
| E'+'E
| E'-'E
| E'*'E
| E'/'E
| E'<'E
| E'>'E
| ID
| NUMBER
         ;
E2      : E'<'E
| E'>'E
| E LE E
| E GE E
| E EQ E
| E NE E
| E OR E
| E AND E
| ID
| NUMBER
;

%%


int yyerror(char *msg){
    printf("\nSyntax error!");
    exit(0);
}

int main(){
printf("Enter the  do...while code : \n");
return yyparse();

}
int yywrap(){

    return 1;
}
```

Output:



```
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> flex E42.l
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> bison -dy E42.y
conflicts: 2 shift/reduce
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> gcc lex.yy.c y.tab.c
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the  do...while code :
do{a=a+1;}while(a<10);

The syntax is correct!
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the  do...while code :
do{a=a+1}while(a<10);

Syntax error!
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> 
```

E5. Write YACC specification to recognize strings that can be accepted by grammar of the form:

a ^n b ^n c, n&gt;=1

File E5.l

```
%{
```

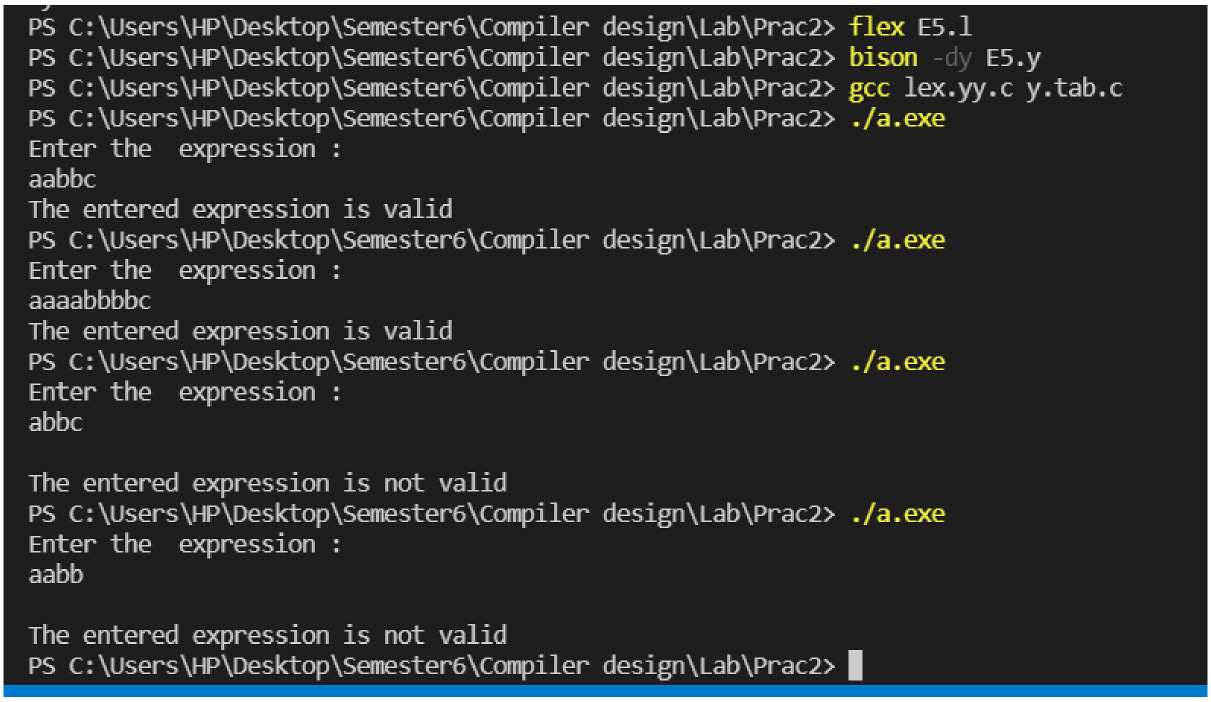
```
#include "y.tab.h"
%}

%%

a   {return A;}
b       {return B;}
c   {return C;}
\n {return NL;}
. {return yytext[0];}

%%
```

File E5.y

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

%}

%token A B C NL
%left '+' '-'
%left '*' '/'

%%
stmt : eval C NL {
    printf("The entered expression is valid");
            exit(0);
}

eval : A eval B
|
;

%%

int yyerror(char *msg){
    printf("\nThe entered expression is not valid");
    exit(0);
}

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

}
```

```
int yywrap(){

    return 1;
}
```

Output



```
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> flex E5.l
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> bison -dy E5.y
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> gcc lex.yy.c y.tab.c
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the  expression :
aabbc
The entered expression is valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the  expression :
aaaabbbbc
The entered expression is valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the  expression :
abbc

The entered expression is not valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the  expression :
aabb

The entered expression is not valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2>
```

E6. Write YACC specification to recognize strings that can be accepted by grammar of the form: {L= a n   b 2n c, n&gt;=1 }

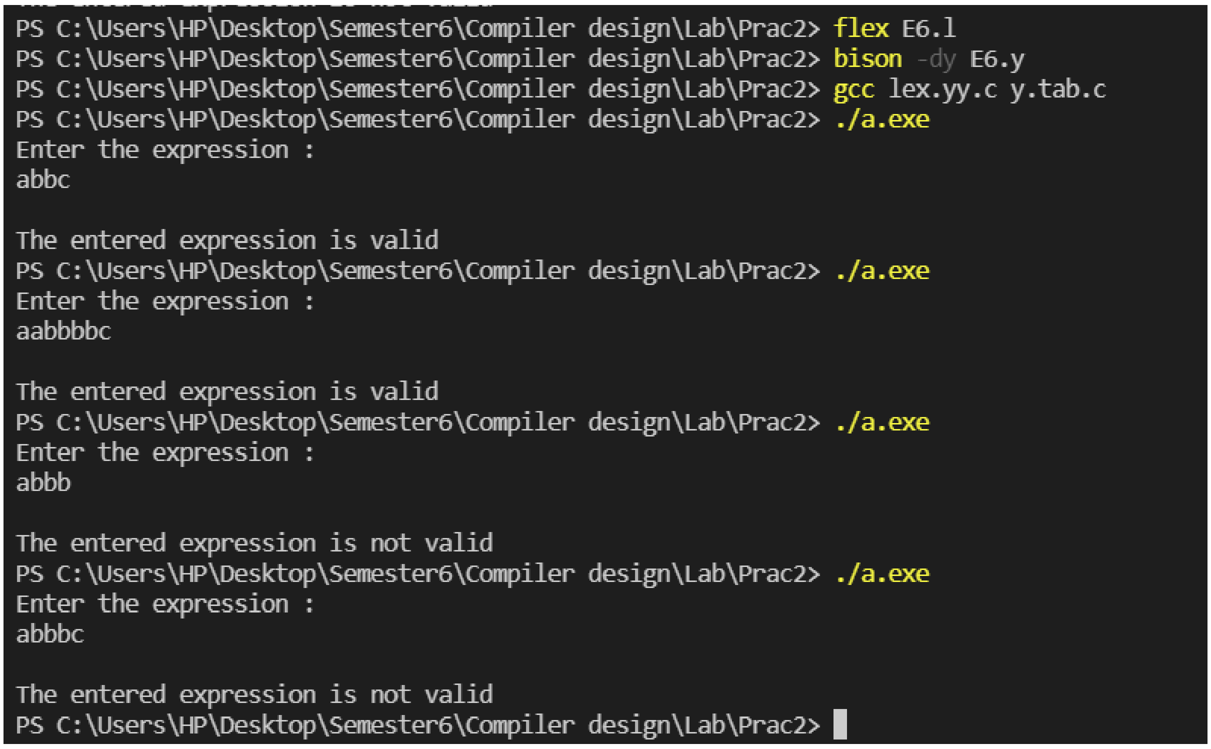
File E6.l

```
%{

#include "y.tab.h"
%}

%%
a    {return A;}
b    {return B;}
c    {return C;}
[\n\t]       {return yytext[0];}
.    {return yytext[0];}
%%
```

File E7.y

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

%}
%token A B C
%left '+' '-'
%left '*' '/'
%%

exp :
 S C '\n' {
    printf("\nThe entered expression is valid");
            exit(0);
}


S : A S val
|
;

val : B B
;

%%
int yyerror(char *msg){
    printf("\nThe entered expression is not valid");
    exit(0);
}

int main(){

printf("Enter the expression : \n");
return yyparse();

}

int yywrap(){

    return 1;
}
```

Output:



```
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> flex E6.l
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> bison -dy E6.y
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> gcc lex.yy.c y.tab.c
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the expression :
abbc

The entered expression is valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the expression :
aabbbbc

The entered expression is valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the expression :
abbb

The entered expression is not valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2> ./a.exe
Enter the expression :
abbbc

The entered expression is not valid
PS C:\Users\HP\Desktop\Semester6\Compiler design\Lab\Prac2>
```