Program :

> **lex**

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

%%
[aA] {return A;}
[bB] {return B;}
\n {return NL;}
. {return yytext[0];}
%%

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

> **yacc**

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

%token A B NL

%%
stmt: S NL { printf("Valid String\n");exit(0);} ;
S: A S B | ;
%%

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

int main()
{
	printf("\nEnter the string : ");
	yyparse();
	return 0;
}
```

Output :

```
:~/Desktop/Lab$ lex 8.l
:~/Desktop/Lab$ yacc -d 8.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter the string : aabb
Valid String
:~/Desktop/Lab$ ./a.out

Enter the string : abab
Invalid String
:~/Desktop/Lab$
```

Program :

> **lex**

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

%%
[aA] {return A;}
[bB] {return B;}
\n {return NL;}
. {return yytext[0];}
%%

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

> **yacc**

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

%token A B NL

%%
stmt: S NL { printf("Valid String\n"); exit(0); };
S: A S | B S | A B B ;
%%

int yyerror(char *msg)
{
        printf("Invalid String\n");
```

```
        exit(0);
}

int main()
{
        printf("\nEnter the string : ");
        yyparse();
        return 0;
}
```

Output :

```
:~/Desktop/Lab$ lex 9.l
:~/Desktop/Lab$ yacc -d 9.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter the string : ab
Invalid String
:~/Desktop/Lab$ ./a.out

Enter the string : abb
Valid String
:~/Desktop/Lab$
```

Program :

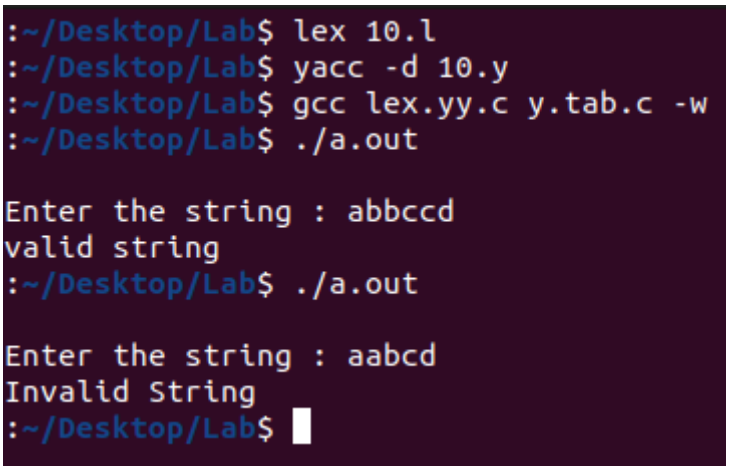
> **lex**

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

%%
[aA] {return A;}
[bB] {return B;}
[cC] {return C;}
[dD] {return D;}
\n {return NL;}
. {return yytext[0];}
%%

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

> **yacc**

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

%token A B C D NL

%%
stmt: S NL { printf("valid string\n");exit(0); } ;
S: A S D | A X D
X: B X C | ;
%%

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

int main()
{
	printf("\nEnter the string : ");
	yyparse();
	return 0;
}
```

Output :



```
:~/Desktop/Lab$ lex 10.l
:~/Desktop/Lab$ yacc -d 10.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter the string : abbccd
valid string
:~/Desktop/Lab$ ./a.out

Enter the string : aabcd
Invalid String
:~/Desktop/Lab$
```

Program :

> **lex**

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

%%
[aA] {return A;}
[bB] {return B;}
[cC] {return C;}
\n {return NL;}
. {return yytext[0];}
%%

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

> **yacc**

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

%token A B C NL

%%
stmt: S NL { printf("valid string\n");exit(0); } ;
S: X Y
X: A X B | A B |
Y: B Y C | B C |  ;
%%

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

int main()
{
	printf("\nEnter the string : ");
	yyparse();
	return 0;
}
```

Output :

```
:~/Desktop/Lab$ ./a.out

Enter the string : aabbbc
valid string
:~/Desktop/Lab$ ./a.out

Enter the string : abbbc
Invalid String
:~/Desktop/Lab$
```

Program :

> **lex**

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

%%
[aA] {return A;}
[bB] {return B;}
[cC] {return C;}
\n {return NL;}
. {return yytext[0];}
%%

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

> **yacc**

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

%token A B C  NL

%%
stmt: S NL { printf("valid string\n");exit(0);} ;
S: A S A | B S B | C ;
%%

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

```
int main()
{
        printf("\nEnter the string : ");
        yyparse();
        return 0;
}
```

Output :

```
:~/Desktop/Lab$ lex 12.l
:~/Desktop/Lab$ yacc -d 12.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter the string : abaacaaba
valid string
:~/Desktop/Lab$ ./a.out

Enter the string : abcab
Invalid String
:~/Desktop/Lab$ ./a.out

Enter the string : abba
Invalid String
:~/Desktop/Lab$
```

Program :

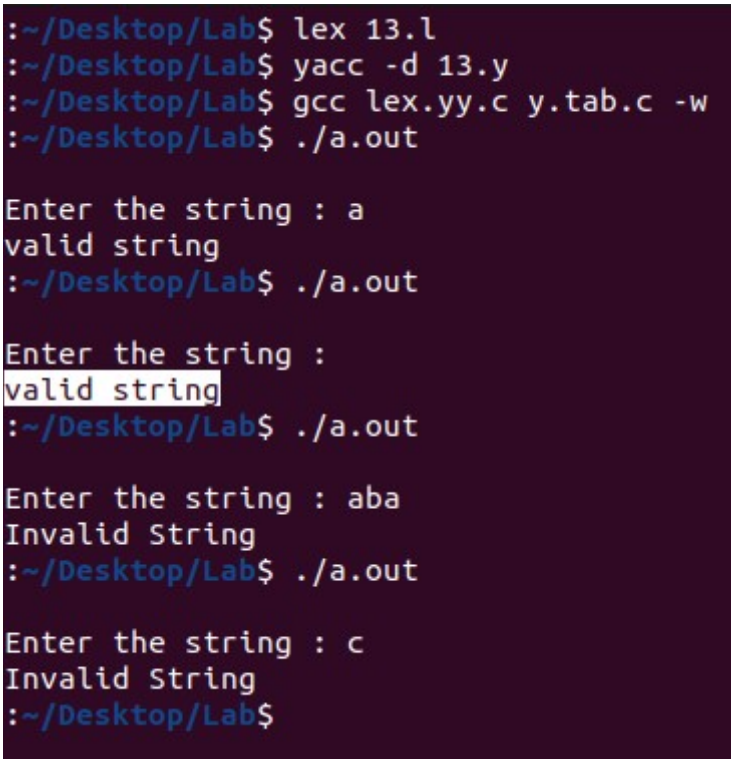
> **<u>lex</u>**

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

%%
[aA] {return A;}
[bB] {return B;}
\n {return NL;}
.  {return yytext[0];}
%%

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

> **yacc**

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

%token A B NL

%%
stmt: S NL {printf("valid string\n");exit(0);} ;
S: A S | X
X: B X | ;
%%

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

int main()
{
	printf("\nEnter the string : ");
	yyparse();
	return 0;
}
```

Output :



```
:~/Desktop/Lab$ lex 13.l
:~/Desktop/Lab$ yacc -d 13.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter the string : a
valid string
:~/Desktop/Lab$ ./a.out

Enter the string :
valid string
:~/Desktop/Lab$ ./a.out

Enter the string : aba
Invalid String
:~/Desktop/Lab$ ./a.out

Enter the string : c
Invalid String
:~/Desktop/Lab$
```

Program :

> **lex**

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

DIGIT [0-9]
REAL {DIGIT}+[.]{DIGIT}*
LETTER [A-Za-z]
ASSIGN =

%%
[\t ] ;
int {printf("%s\t==> DataType\n",yytext);return (INT);}
float {printf("%s\t==> DataType\n",yytext);return (FLOAT);}
char {printf("%s\t==> DataType\n",yytext);return (CHAR);}
boolean {printf("%s\t==> DataType\n",yytext);return (BL);}
true|false { printf("%s\t==> BOOLEAN VAL\n",yytext);return BLVAL;}
']['^\t\n']['] { printf("%s\t==> CHAR VALUE\n",yytext);return CHVAL;}
[a-zA-z]+[a-zA-z0-9_]*  {printf("%s\t==> ID\n",yytext);return ID;}
{REAL} { printf("%s\t==> REAL NUMBER\n",yytext);return REAL;}
{DIGIT}+ { printf("%s\t==> INT NUMBER\n",yytext);return NUM;}
"," {printf("%s\t==> COMMA\n",yytext);return COMMA;}
";" {printf("%s\t==> SC\n",yytext);;return SC;}
{ASSIGN} {printf("%s\t==> ASSIGN\n",yytext);return AS;}
\n {return NL;}
. ;
%%

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

> **yacc**

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

%token ID SC INT CHAR FLOAT BL BLVAL CHVAL REAL AS NUM COMMA NL

%%
s: type1|type2|type3|type4 ;
type1: INT varlist SC NL { printf("valid INT Variable\n "); return 0;} ;
type2: FLOAT varlist2 SC NL{ printf("valid FLOAT Variable declaration\n");return 0;} ;
type3: CHAR varlist3 SC NL{ printf("valid CHAR Variable declaration\n");return 0;} ;
type4: BL varlist4 SC NL{ printf("valid BOOLEAN Variable declaration\n");return 0;} ;
varlist:  ID | ID COMMA varlist | ID AS NUM |ID AS NUM COMMA varlist | ;
varlist2:  ID | ID COMMA varlist2 | ID AS REAL |ID AS REAL COMMA varlist2 | ;
```

```
varlist3: ID | ID COMMA varlist3 | ID AS CHVAL | ID AS CHVAL COMMA varlist3 | ;
varlist4: ID | ID COMMA varlist4 | ID AS BLVAL | AS BLVAL COMMA varlist4 | ;
%%

void yyerror(char *s )
{
	fprintf(stderr, "ERROR: %s\n",s);
}

int main()
{
	printf("\nEnter the string : ");
	yyparse();
	return 0;
}
```

Output :

```
:~/Desktop/Lab$ lex 14.l
:~/Desktop/Lab$ yacc -d 14.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter the input : int a
int      ==> DataType
a        ==> ID
ERROR: syntax error
:~/Desktop/Lab$ ./a.out

Enter the input : int a;
int      ==> DataType
a        ==> ID
;        ==> SC
valid INT Variable
 :~/Desktop/Lab$
```

Program :

> **lex**

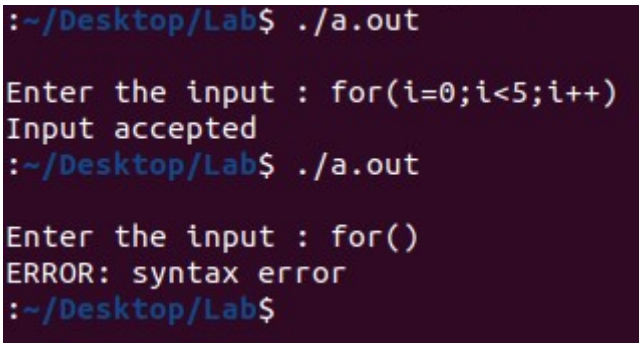
```
%{
#include <stdio.h>
#include "y.tab.h"
%}

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

%%
[\t \n]
for return FOR;
{digit}+ return NUM;
{alpha}({alpha}|{digit})* return ID;
"<=" return LE;
">=" return GE;
"==" return EQ;
"!=" return NE;
"||" return OR;
"&&" return AND;
. return yytext[0];
%%

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

> **yacc**

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

%token ID NUM FOR LE GE EQ NE OR AND

%%

S : ST {printf("Input accepted\n"); exit(0);} ;
ST : FOR '(' E ';' E2 ';' E ')' ;
E : ID '=' E
   | E '+' E
   | E '-' E
   | E '*' E
   | E '/' E
   | E '<' E
   | E '>' E
   | E LE E
   | E GE E
   | E EQ E
   | E NE E
```

```
    | E OR E
    | E AND E
    | E '+' '+'
    | E '-' '-'
    | ID
    | NUM | ;
E2 : E'<'E
     | E'>'E
     | E LE E
     | E GE E
     | E EQ E
     | E NE E
     | E OR E
     | E AND E | ;
%%

void yyerror(char *s)
{
        fprintf(stderr, "ERROR: %s\n",s);
}

int main()
{
        printf("\nEnter the input : ");
        yyparse();
        return 0;
}
```

Output :



Program :

> **lex**

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

%%
void|int|float|char {return BUILTIN;}
, {return COMMA;}
; {return SC;}
\( {return LP;}
\) {return RP;}
[a-zA-Z0-9 ]* {return ID;}
\n {return NL;}
. ;
%%

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

> **yacc**

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%token COMMA ID BUILTIN SC NL RP LP
%%
var: datatype varlist LP FUNCT RP SC NL  {printf("Valid declaration\n");return 0;};
|
datatype: BUILTIN ;
|
varlist: ID ;
|
FUNCT: FUNCT COMMA ID|FUNCT COMMA BUILTIN|BUILTIN|ID| ;
%%

void yyerror(char *s)
{
	fprintf(stderr, "ERROR: %s\n",s);
}

int main()
{
	printf("\nEnter the input : ");
	yyparse();
	return 0;
}
```

Output :

```
:~/Desktop/Lab$ lex 16.l
:~/Desktop/Lab$ yacc -d 16.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter the input : int main();
Valid declaration
:~/Desktop/Lab$ ./a.out

Enter the input : int main()
ERROR: syntax error
:~/Desktop/Lab$ ./a.out

Enter the input : void sum(int a);
Valid declaration
:~/Desktop/Lab$
```

Program :

> **lex**

```
%{
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
%}

%%
[0-9]+ {yylval=atoi(yytext);return NUMBER;}
[\t] ;
[\n] {return 0;}
. {return yytext[0];}
%%

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

> **yacc**

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

%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
```

```
%%
ArithmeticExpression: E{printf("\nResult = %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 Any Arithmetic Expression which can have operations
Addition,Subtraction,Multiplication, Division, Modulus and Round brackets:\n");
  yyparse();
  if(flag==0)
  printf("\nEntered arithmetic expression is Valid \n ");
}

int yyerror()
{
  printf("\nEntered arithmetic expression is Invalid\n\n");
  flag=1;
}
```

Output :

```
:~/Desktop/Lab$ lex 17.l
:~/Desktop/Lab$ yacc -d 17.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition,Subtraction,Multiplication, Division, Modulus and Round brackets:
50+3

Result = 53

Entered arithmetic expression is Valid
 :~/Desktop/Lab$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition,Subtraction,Multiplication, Division, Modulus and Round brackets:
56-3

Result = 53

Entered arithmetic expression is Valid
 :~/Desktop/Lab$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition,Subtraction,Multiplication, Division, Modulus and Round brackets:
4*2

Result = 8

Entered arithmetic expression is Valid
 :~/Desktop/Lab$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition,Subtraction,Multiplication, Division, Modulus and Round brackets:
4/2

Result = 2

Entered arithmetic expression is Valid
 :~/Desktop/Lab$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition,Subtraction,Multiplication, Division, Modulus and Round brackets:
3%2

Result = 1

Entered arithmetic expression is Valid
 :~/Desktop/Lab$
```

Program :

> **lex**

```
%{
#include"y.tab.h"
extern int yylval;
%}

%%
[0-9]+ {yylval=atoi(yytext); return NUM;}
\n {return 0;}
. {return *yytext;}
%%

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

> **yacc**

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

%token NUM
%left '+' '-'
%left '*' '/'
%right NEGATIVE

%%
S: E {printf("\n");} ;
E: E '+' E {printf("+");}
   | E '*' E {printf("*");}
   | E '-' E {printf("-");}
   | E '/' E {printf("/");}
   | '(' E ')'
   | '-' E %prec NEGATIVE {printf("-");}
   | NUM {printf("%d", yylval);};
%%

void yyerror(char *s)
{
	fprintf(stderr, "ERROR: %s\n",s);
}

int main()
{
	printf("\nEnter the input : ");
	yyparse();
	return 0;
}
```

Output :

```
:~/Desktop/Lab$ lex 18.l
:~/Desktop/Lab$ yacc -d 18.y
:~/Desktop/Lab$ gcc lex.yy.c y.tab.c -w
:~/Desktop/Lab$ ./a.out

Enter the input : 2*4+8
24*8+
:~/Desktop/Lab$
```