

Ex No: 5  
Date:

## RECOGNIZE AN ARITHMETIC EXPRESSION USING LEX AND YACC

### AIM:

To check whether the arithmetic expression using lex and yacc tool.

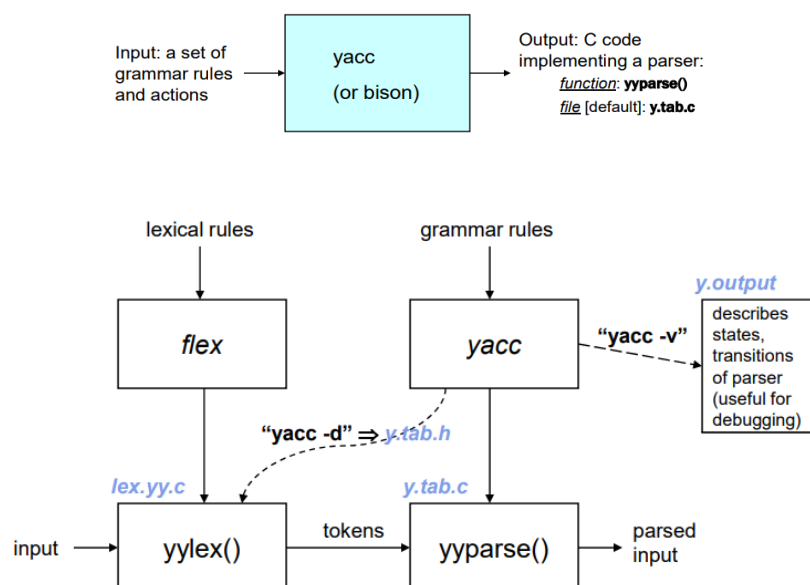
### ALGORITHM:

- Using the flex tool, create lex and yacc files.
- In the C include section define the header files required.
- In the rules section define the REGEX expressions along with proper definitions.
- In the user defined section define yywrap() function.
- Declare the yacc file inside it in the C definitions section declare the header files required along with an integer variable valid with value assigned as 1.
- In the Yacc declarations declare the format token num id op.
- In the grammar rules section if the starting string is followed by assigning operator or identifier or number or operator followed by a number or open parenthesis followed by an identifier. The x could be an operator followed by an identifier or operator or no operator then declare that as valid expressions by making the valid stay in 1 itself.
- In the user definition section if the valid is 0 print as Invalid expression in yyerror() and define the main function.

### LEX AND YACC WORKING :

Parser generator:

- Takes a specification for a context-free grammar.
- Produces code for a parser.



**PROGRAM:****validexp.l:**

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

%%
[a-zA-Z]+ return VARIABLE;
[0-9]+ return NUMBER;
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

**validexp.y:**

```
% {
#include<stdio.h>
% }
%token NUMBER
%token VARIABLE

%left '+' '-'
%left '*' '/' '%'
%left '(' ')'

%%

S: VARIABLE='E' {
    printf("\nEntered arithmetic expression is Valid\n\n");
    return 0;
}
E:E'+E
|E'-E
|E'*E
|E'/E
|E'%E
|('E')
|NUMBER
|VARIABLE
;
```

```
%%
```

```
void main()
{
    printf("\nEnter Any Arithmetic Expression which can have operations Addition,
Subtraction, Multiplication, Divison, Modulus and Round brackets:\n");
    yyparse();
}

void yyerror()
{
    printf("\nEntered arithmetic expression is Invalid\n\n");
}

}
```

### OUTPUT:

```
[root@localhost-live 210701265]m vi ex5.c
[root@localhost-live 216701265]# vi ex5.l
[root@localhost-live 210701265]# ty ex5.y
[root@localhost-live 210701265]n lex ex5.1
[root@localhost-live 210701265]# yacc -d ex5.y
[root@localhost-live 218701265]# cc lex.yy.c y.tab.c
[root@localhost-live 218701265]# ./a.out

Enter Any Arithmetic Expression which can have operations Addition,Subtraction, Multiplication, Divison, Modulus and Round brackets:
24-27

Entered arithmetic expression is Invalid

[root@localhost-live 219701265]# ./a.out

Any Arithmetic Expression which can have operations Addition,Subtraction, Multiplication, Divison, Modulus and Round brackets:
a=2+3

Entered arithmetic expression is Valid
```

### RESULT: