Ex No: 5
Date:

RECOGNIZE AN ARITHMETIC EXPRESSION USING LEX AND YACC

AIM:

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

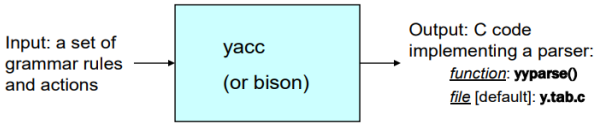
ALGORITHM:

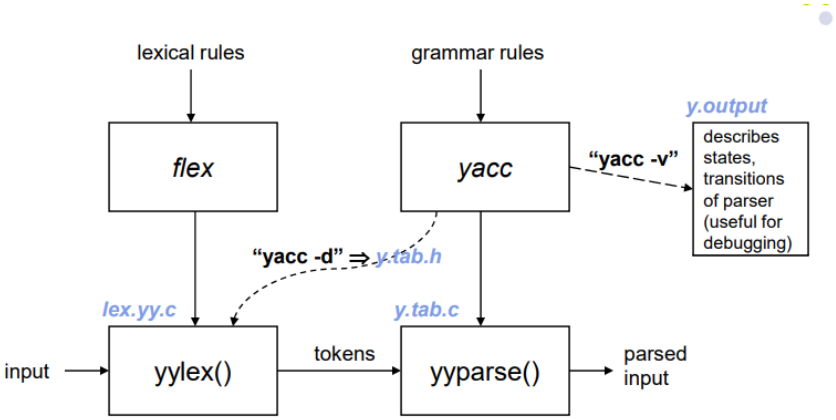
- Using the flex tool, create lex and yacc files.
- In the C include section define the header files required.
- In the rules section define the REGEX expressions along with proper definitions.
- In the user defined section define yywrap() function.
- Declare the yacc file inside it in the C definitions 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:

arithmetic303.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;
}
```

arithmetic303.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:

```
[VaruneshB210701303@localhost ~]$ vi arithmetic303.c
[VaruneshB210701303@localhost ~]$ vi arithmetic303.l
[VaruneshB210701303@localhost ~]$ vi arithmetic303.y
[VaruneshB210701303@localhost ~]$ lex arithmetic303.l
[VaruneshB210701303@localhost ~]$ yacc -d arithmetic303.y
[VaruneshB210701303@localhost ~]$ cc lex.yy.cy.tab.c
[VaruneshB210701303@localhost ~]$ ./a.out
Enter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brackets:
303+303

Entered arithmetic expression is Invalid
[VaruneshB210701303@localhost ~]$ ./a.out
Enter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brackets:
z=100+120

Entered arithmetic expression is Valid
[VaruneshB210701303@localhost ~]$
```

RESULT: