Department of CSE
SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VI

7 March 2021

UCS 1602 - Compiler Design

Exercise 5: Implementation of Desk Calculator Using Yacc Tool

Aim:

Write a Lex program to recognize relevant tokens required for the **Yacc** parser to implement desk calculator. Write the Grammar for the expression involving the operators namely, $+, -, *, /, \hat{}\ , (,)$. Precedence and associativity has to be preserved. Yacc is available as a command in Linux. The grammar should have non terminals E, OP and a terminal id.
Verify your calculator with the following inputs:

1. $3 + 9$
2. $3 + 9 * 6$
3. $(3 + 4) * 7$
4. $(3 - 4) + (7 * 6)$
5. $5/7 + 2$
6. $(4^2)^1$
7. $(2^3)^2$

Code - Yacc Parser File:

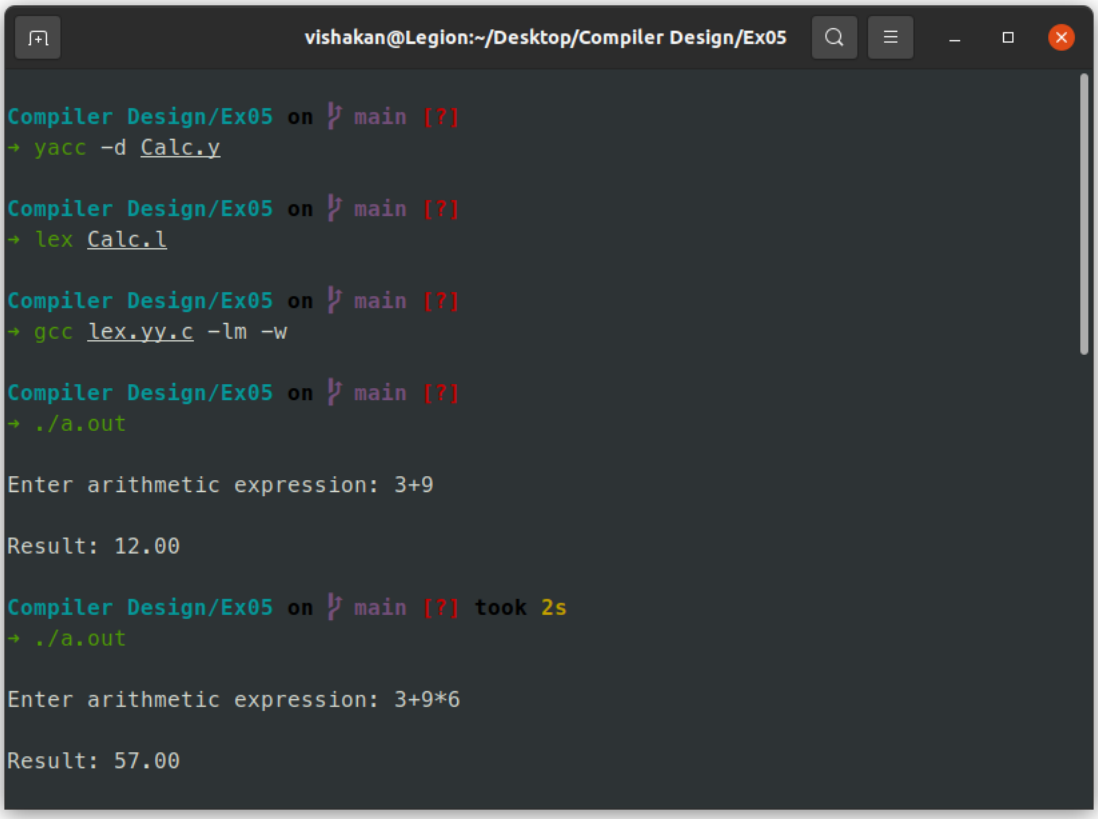
```
%{
   #include <stdio.h>
   #include <math.h>
   #define YYSTYPE double
   int flag = 0;
%}

%token NUM
/*Defining the precedence*/
%left '+' '-'
%left '/' '*'
%right '^'
%left '(' ')'

%%
Line  :  Expr            {printf("\nResult: %.2f\n", $$);}
Expr  :  Expr '+' Expr   {$$ = $1 + $3;}
      |  Expr '-' Expr   {$$ = $1 - $3;}
      |  Expr '*' Expr   {$$ = $1 * $3;}
      |  Expr '/' Expr   {$$ = $1 / $3;}
      |  Expr '^' Expr   {$$ = pow($1, $3);}
      |  '('Expr')'      {$$ = $2;}
      |  NUM             {$$ = $1;}
%%

int yyerror(){
   flag = 1;
   return 1;
}

int main(void){
   printf("\nEnter arithmetic expression: ");
   yyparse();

   if(flag){
      printf("\nEntered Unexpected Tokens.\n");
   }

   return 0;
}

/* Usage:
         Run yacc -d Calc.y
         Run lex Calc.l
         Run gcc lex.yy.c -lm -w
         Run ./a.out
*/
```

Code - Lex Grammar File:

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

%%

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

%%

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

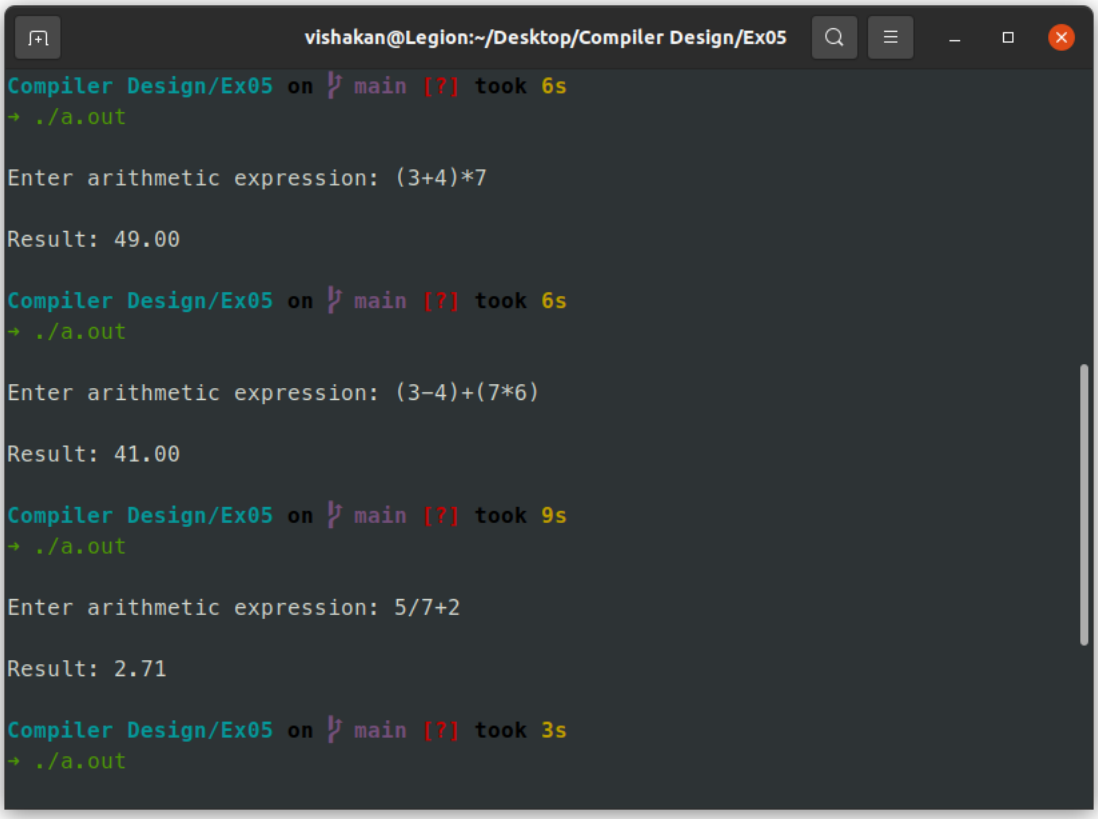
Output 1:

Figure 1: Console Output - 1.



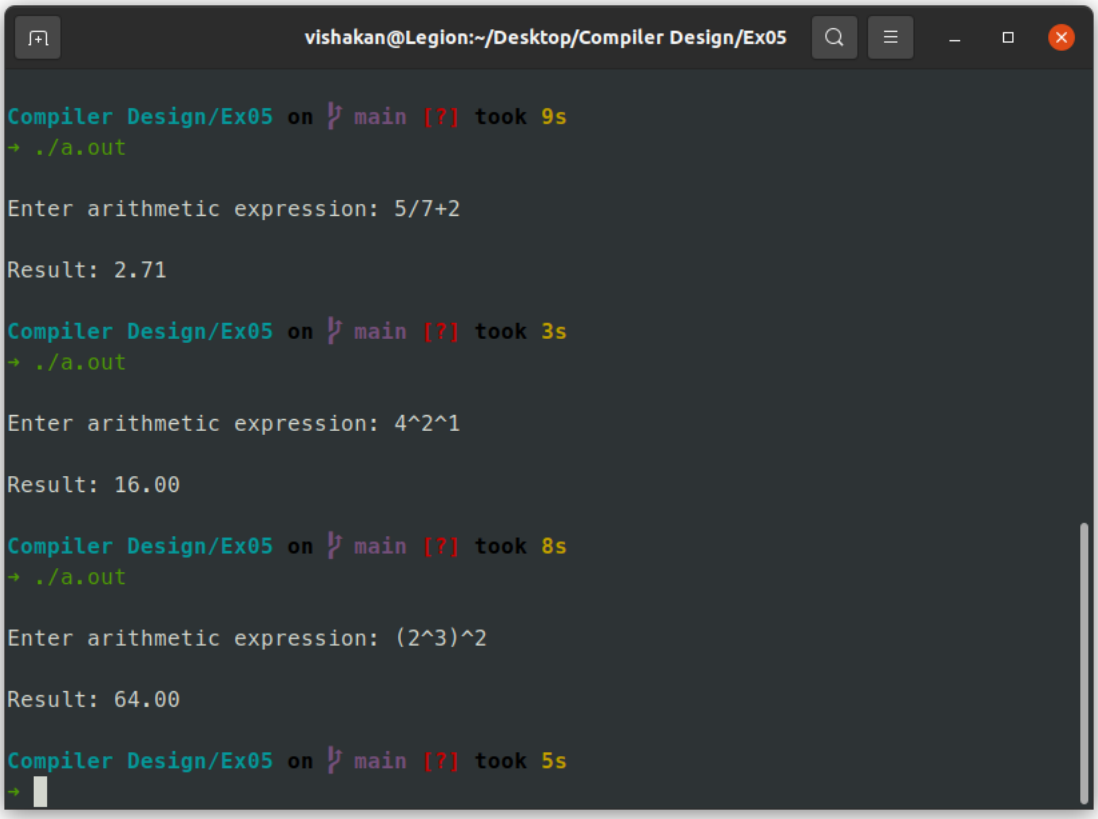
Output 2:

Figure 2: Console Output - 2.



Output 3:

Figure 3: Console Output - 3.



Learning Outcome:

- I learnt the basic theory behind **Yacc Parser Generator**.
- I learnt that Yacc stands for Yet Another Compiler-Compiler.
- I understood that Yacc is a LALR(1) parser.
- I understood Yacc's basic syntax and programming logic.
- I learnt that Yacc needs a Lex file along with it to work as intended, to detect and give the tokens to the Yacc parser.
- I was able to visualize how the parser works with the scanner.
- I learnt how to define a simple grammar in Yacc's syntax.
- I was able to implement a parser with Yacc to mimic the features of a desk calculator with precedence logic.
- I understood how to compile and run the Yacc and Lex file together.