Ex-no: 4 M. Rohith
08-11-2024 3122 21 5001 085

**SSN COLLEGE OF ENGINEERING,
KALAVAKKAM
DEPARTMENT OF COMPUTER SCIENCE
& ENGINEERING**

UCS 2702- Compiler Design Lab
Exercise 4 – Implementation of Desk Calculator using Yacc Tool

Write 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, + , - ,* , / , ^ , (,). **Precedence and associativity has to be preserved.**

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

Tips to use tools

Write Lex specification, compile and execute to check for the tokens, namely, operators and the identifiers.
Write yacc specification in ex.y and type the command yacc ex.y.The output will be y.tab.c
Compile using the command cc y.tab.c. The output will be a.out
Use exe to give input and get the output.

Program code:

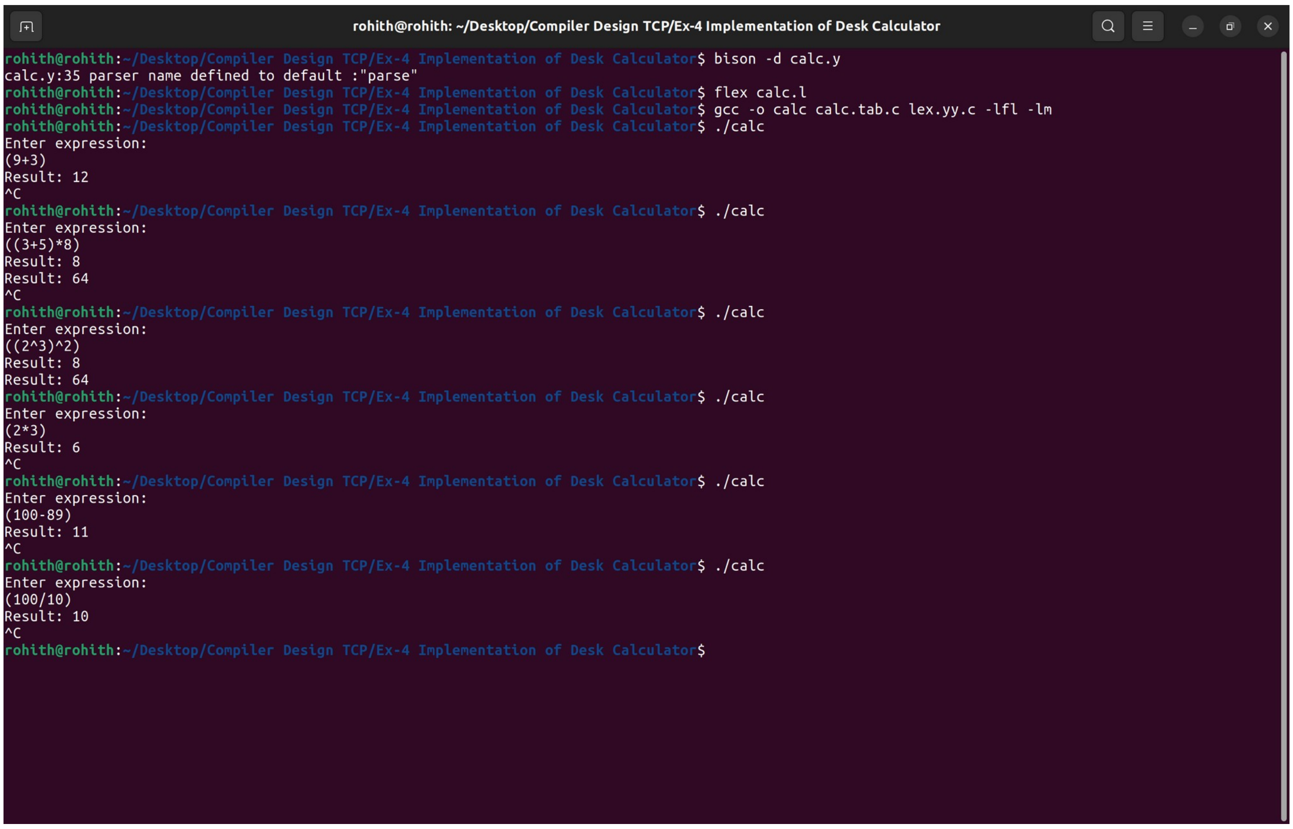
calc.l

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

%%
[0-9]+     { yylval = atoi(yytext); return NUMBER; }
"+"        { return '+'; }
"-"        { return '-'; }
"*"        { return '*'; }
"/"        { return '/'; }
"^"        { return '^'; }
"("        { return '('; }
")"        { return ')'; }
[ \t\n]+   { /* ignore whitespace */ }
.          { printf("Unknown character: '%s'\n", yytext); }

%%

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

calc.y

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

// Function prototypes
int yylex(void);    // Declaration for yylex to avoid implicit declaration warning
void yyerror(const char *s);
%}

%token NUMBER

%left '+' '-'
%left '*' '/'
%right '^'

%%
// Grammar rules
expr: expr '+' expr { $$ = $1 + $3; printf("Result: %d\n", $$); }
     | expr '-' expr { $$ = $1 - $3; printf("Result: %d\n", $$); }
     | expr '*' expr { $$ = $1 * $3; printf("Result: %d\n", $$); }
     | expr '/' expr {
            if ($3 == 0) {
                yyerror("Error: Division by zero!");
                $$ = 0; // Return zero in case of error
            } else {
                $$ = $1 / $3;
                printf("Result: %d\n", $$);
            }
          }
      | expr '^' expr  { $$ = pow($1, $3); printf("Result: %d\n", $$); }
      | '(' expr ')'   { $$ = $2; }
      | NUMBER         { $$ = $1; }
      ;
%%
void yyerror(const char *s) {
    fprintf(stderr, "%s\n", s);
}

int main() {
    printf("Enter expression:\n");
    while (1) {
        yyparse();
    }
    return 0;
}
```

Output:


rohith@rohith: ~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator

```
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ bison -d calc.y
calc.y:35 parser name defined to default :"parse"
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ flex calc.l
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ gcc -o calc calc.tab.c lex.yy.c -lfl -lm
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ ./calc
Enter expression:
(9+3)
Result: 12
^C
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ ./calc
Enter expression:
((3+5)*8)
Result: 8
Result: 64
^C
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ ./calc
Enter expression:
((2^3)^2)
Result: 8
Result: 64
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ ./calc
Enter expression:
(2*3)
Result: 6
^C
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ ./calc
Enter expression:
(100-89)
Result: 11
^C
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$ ./calc
Enter expression:
(100/10)
Result: 10
^C
rohith@rohith:~/Desktop/Compiler Design TCP/Ex-4 Implementation of Desk Calculator$
```