LEX FILE: CAL.L

```
number [0-9]+\.?|[0-9]*\.[0-9]+
%%
[ \t]   { ; }
log     return LOG;
factorial return FACTORIAL;
pi      return PIVAL;
sin     return SIN;
cos     return COS;
tan     return TAN;
cot     return COT;
sec     return SEC;
cosec   return COSEC;
sinh    return SINH;
cosh    return COSH;
tanh    return TANH;
coth    return COTH;
sech    return SECH;
cosech  return COSECH;
asin    return ASIN;
acos    return ACOS;
atan    return ATAN;
acot    return ACOT;
asec    return ASEC;
acosec   return ACOSEC;
log10  return LOG10;
{number}        { yylval=atof(yytext);return NUMBER; }
"++"    return INC;
"--"    return DEC;
```

```
"+"     return PLUS;
"-"     return MINUS;
"~"     return UNARYMINUS;
"/"     return DIV;
"*"     return MUL;
"^"     return POW;
sqrt    return SQRT;
"("     return OPENBRACKET;
")"     return CLOSEBRACKET;
"%"     return MOD;
\n      return 0;
.       return yytext[0];
```

YACC FILE: CAL.Y

```
%{
#include<stdio.h>
#include <math.h>
#define YYSTYPE double
float factorial(int n)
{
        if (n == 1)
        return 1;
        return (n * factorial(n-1));
}
%}

%token NUMBER MOD PI
%token PLUS MINUS DIV MUL POW SQRT OPENBRACKET CLOSEBRACKET
UNARYMINUS
%token SIN COS TAN COT SEC COSEC SINH COSH TANH COTH SECH COSECH ASIN
ACOS ATAN ACOT ASEC ACOSEC INC DEC FACTORIAL
```

```
%left PLUS MINUS MUL DIV UNARYMINUS LOG LOG10
%%

expr: add
      | pow POW add { $$ = pow($1,$3); }
        | SQRT OPENBRACKET expr CLOSEBRACKET { $$ = sqrt($3) ; }
      ;
add: mul
      | add PLUS mul  { $$ = $1 + $3;}
      | add MINUS mul { $$ = $1 - $3; }
      ;
mul: unary
      | mul MUL unary { $$ = $1 * $3; }
      | mul DIV unary { $$ = $1 / $3; }
      | mul MOD unary { $$ = fmod($1,$3); }
      ;
unary: post
      | MINUS primary %prec UNARYMINUS { $$ = -$2; }
      | INC unary { $$ = $2+1; }
      | DEC unary { $$ = $2-1; }
      | LOG unary { $$ = log($2); }
      | LOG10 unary { $$ = log10($2); }
      ;
post   : primary
      | post INC { $$ = $1+1; }
      | post DEC { $$ = $1-1; }
      ;
primary:
       PIVAL { $$ = M_PI; }
      | OPENBRACKET expr CLOSEBRACKET { $$ = $2; }
      | function
      ;
function: SIN OPENBRACKET expr CLOSEBRACKET
```

```
    { $$ = sin($3); }
| COS OPENBRACKET expr CLOSEBRACKET
    { $$ = cos($3); }
| TAN OPENBRACKET expr CLOSEBRACKET
    { $$ = tan($3); }
| COT OPENBRACKET expr CLOSEBRACKET
    { $$ = 1/tan($3); }
| SEC OPENBRACKET expr CLOSEBRACKET
    { $$ = 1/cos($3); }
| COSEC OPENBRACKET expr CLOSEBRACKET
    { $$ = 1/sin($3); }
  | SINH OPENBRACKET expr CLOSEBRACKET
    { $$ = sinh($3); }
| COSH OPENBRACKET expr CLOSEBRACKET
    { $$ = cosh($3); }
| TANH OPENBRACKET expr CLOSEBRACKET
    { $$ = tanh($3); }
| COTH OPENBRACKET expr CLOSEBRACKET
    { $$ = 1/tanh($3); }
| SECH OPENBRACKET expr CLOSEBRACKET
    { $$ = 1/cosh($3); }
| COSECH OPENBRACKET expr CLOSEBRACKET
    { $$ = 1/sinh($3); }
  | ASIN OPENBRACKET expr CLOSEBRACKET
    { $$ = asin($3); }
| ACOS OPENBRACKET expr CLOSEBRACKET
    { $$ = acos($3); }
| ATAN OPENBRACKET expr CLOSEBRACKET
    { $$ = atan($3);}
  | ACOT OPENBRACKET expr CLOSEBRACKET
    { $$ = atan(1/$3); }
| ASEC OPENBRACKET expr CLOSEBRACKET
```

```
            { $$ = acos(1/$3); }
        | ACOSEC OPENBRACKET expr CLOSEBRACKET
            { $$ = asin(1/$3); }
          | FACTORIAL OPENBRACKET expr CLOSEBRACKET
                { $$ = factorial((int)$3);}
          | NUMBER
        ;
%%

#include <stdio.h>
#include "lex.yy.c"
void yyerror(const char *s)
 {
 fprintf(stderr, "%s\n", s);
 }
int main(void)
 {       while(1){
         yyparse();
         }
 return 0;
 }
```