```
runner% cat eval1.c
/* arith.c: simple parser -- bold  evaluates */
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>  /* for pow */
char next;
double E(void);
double T(void);
double S(void);
double F(void);
void error(void);
void scan(void);
double evaluate(double arg1, char op, double arg2);

void main(void)
{
   double res;
   scan();
   res = E();
   if (next != '#') error();
   else
      printf("Result is: %.5f\n", res);
}

double E(void)
{
   char save;
   double res, arg1, arg2;
   res = T();
   while (next == '+' || next == '-') {
      save = next;
      arg1 = res;
      scan();
      arg2 = T();
      res = evaluate(arg1, save, arg2);
   }
   return res;
}

double T(void)
{
   char save;
   double res, arg1, arg2;
   res = S();
   while (next == '*' || next == '/') {
      save = next;
      arg1 = res;
      scan();
      arg2 = S();
      res = evaluate(arg1, save, arg2);
   }
   return res;
}

double S(void)
{
   char save;
   double res, arg1, arg2;
   res = F();
   if (next == '^') {
      save = next;
      arg1 = res;
      scan();
      arg2 = S();
      res = evaluate(arg1, save, arg2);
   }
   return res;
}

double F(void)
{
   char save;
   double res;
   if (isdigit(next)) {
      save = next;
      scan();
      return (double)(save - '0');
   }
   else if (next == '(') {
      scan();
      res = E();
      if (next == ')') scan();
      else error();
      return res;
   }
   else {
      error();
      return 0;
```

```
   }
}
void scan(void)
{
   while (isspace(next = getchar()))
      ;
}
void error(void)
{
   printf("\n*** ERROR ***\n");
   exit(1);
}
double evaluate(double arg1, char op, double arg2)
{
   switch (op) {
   case '+': return arg1 + arg2;
   case '-': return arg1 - arg2;
   case '*': return arg1 * arg2;
   case '/': return arg1 / arg2;
   case '^': return pow(arg1, arg2);
   default: error(); return 0;
   }
}
runner% cc -o eval1 eval1.c -lm
runner% eval1
2+3*4#
Result is: 14.00000
runner% eval1
3*4+5#
Result is: 17.00000
runner% eval1
(2+3)*4#
Result is: 20.00000
runner% eval1
(3*(2+4)/(5+1))-2#
Result is: 1.00000
runner% eval1
(5+3)^(2+1)^2#
Result is: 134217728.00000
runner% eval1
2+3*4^5*6+7#
Result is: 18441.00000
runner% eval1
2 + (3*(4^5)*6) + 7#          (same with parentheses)
Result is: 18441.00000
runner% eval1
((3^2-4*1*2)^(1/2)-3)/(2*1)#
Result is: -1.00000
runner% eval1
((2-3)^((4+1)*5)/6-(2-4)*7)-8#
Result is: 5.83333
runner% eval1
3^4^2#
Result is: 43046721.00000
runner% eval1
3^(4^2)#                      (same with parentheses)
Result is: 43046721.00000
runner% eval1
(3^4)^2#
Result is: 6561.00000
runner% eval1
9 - 5 - 3 #
Result is: 1.00000
runner% eval1
8/3/5#
Result is: 0.53333
runner% eval1
8/(3/5)#
Result is: 13.33333
runner% eval1
1+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1
/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2+1/(2
))))))))))))))))))))#
Result is:     1.4142135623730949
(Note:sqrt(2) = 1.4142135623730950488016887242209...)
runner% eval1
2+1/(1+1/(2+1/(1+1/(1+1/(4+1/(1+1/(1+1/(6+1/(1+1/(1+1
/(8+1/(1+1/(1+1/(2*5+1/(1+1/(1+1/(2*6+1/(1+1/(1+1/(2*
7+1  ))))))))))))))))))))#
Result is: 2.7182818284590455
(Note:e =  2.7182818284590452353602874713527...)
runner% eval1
3+1/(7+1/(5*3+1/(1+1/((4*(9*8+1))  ))))#
Result is: 3.1415926530119025
runner% eval1
3+1/(7+1/(5*3+1/(1+1/((4*(9*8+1))+1/(1+1/(1+1/(1+1/(2
+1/(1+1/(3+1/(1+1/(2*7+1/(2+1) ))))))))))))#
Result is: 3.1415926535897931
(Note:pi = 3.14159265358979323846264338327 9...)
```