/* arith0.c: simple parser -- no output

grammar:
  P ---> E '#'
  E ---> T {('+'|'-') T}
  T ---> S {('*'|'/') S}
  S ---> F '^' S | S
  F ---> char | '(' E ')'
*/

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

char next;

void E(void);
void T(void);
void S(void);
void F(void);
void error(void);
void scan(void);
void enter(char);
void leave(char);
void spaces(int);

int level = 0;

int main(void)
{
    scan();
    E();
    if (next != '#') error();
    else printf("Successful parse\n");
}

void E(void)
{
    enter('E');
    T();
    while (next == '+' || next == '-'): scan();
    T();
    leave('E');
}

void T(void)
{
    enter('T');
    S();
    while (next == '*' || next == '/'): scan();
    S();
    leave('T');
}

void S(void)
{
    enter('S');
    F();
    if (next == '^'): scan();
    S();
    leave('S');
}

void F(void)
{
    enter('F');
    if (isalpha(next)): scan();
    else if (next == '('): scan();
    E();
    if (next == ')'): scan();
    else error();
    else error();
    leave('F');
}

void scan(void)
{
    while (isspace(next = getchar()))
}

void error(void)
{
    printf("\n*** ERROR ***\n");
    exit(1);
}

void enter(char name)
{
    printf("%c: Enter, \tNext == %c\n", name, next);
}

void leave(char name)
{
    printf("%c: Leave, \tNext == %c\n", name, next);
}

void spaces(int level)
{
    while (isspace(next = getchar()))
}

int main(void)
{
```c
void spaces(int local_level)
{
    while (local_level-- > 0)
        printf(" ");
}

canada

runner% cc -o arith arith.c
runner% arith
a+b#
E: Enter,      Next == a
T: Enter,      Next == a
S: Enter,      Next == a
F: Enter, Next == a
F: Leave, Next == +
S: Leave, Next == +
T: Leave,      Next == +
T: Enter,      Next == b
S: Enter,      Next == b
F: Enter,    Next == b
F: Leave, Next == #
S: Leave,     Next == #
T: Leave,      Next == #
E: Leave,       Next == #
Successful parse
runner%

runner% arith
da+b+c#
E: Enter,       Next == a
T: Enter,      Next == a
S: Enter,     Next == a
F: Enter,    Next == a
F: Leave,    Next == *
S: Leave,     Next == *
S: Enter,     Next == ( 
F: Enter,    Next == ( 
E: Enter,  Next == b
T: Enter,  Next == b
S: Enter,         Next == b
F: Enter,        Next == b
F: Leave,        Next == +
S: Leave,         Next == +
T: Leave,  Next == +
T: Enter,  Next == c
S: Enter,         Next == c
F: Enter,        Next == c
F: Leave,        Next == )
S: Leave,         Next == )
T: Leave,      Next == )
E: Leave,       Next == )
Successful parse
runner%

runner% arith
da-b-c#
E: Enter,       Next == a
T: Enter,      Next == a
S: Enter,     Next == a
F: Enter,    Next == a
F: Leave,    Next == -
S: Leave,     Next == -
T: Leave,      Next == -
T: Enter,      Next == b
S: Enter,     Next == b
F: Enter,    Next == b
F: Leave,    Next == -
S: Leave,     Next == -
T: Leave,      Next == -
T: Enter,      Next == c
S: Enter,     Next == c
F: Enter,    Next == c
F: Leave,    Next == #
S: Leave,     Next == #
S: Leave,     Next == #
T: Leave,      Next == #
E: Leave,       Next == #
Successful parse
runner%

runner% arith
\ a+b+c#
E: Enter,       Next == a
T: Enter,      Next == a
S: Enter,     Next == a
F: Enter,    Next == a
F: Leave,    Next == ^
S: Leave,     Next == ^
S: Enter,     Next == b
F: Enter,   Next == b
F: Leave,   Next == ^
S: Leave,   Next == ^
S: Enter,   Next == c
F: Enter,  Next == c
F: Leave,  Next == #
S: Leave,   Next == #
S: Leave,    Next == #
S: Leave,     Next == #
T: Leave,      Next == #
E: Leave,       Next == #
Successful parse
runner%