```
/* Reverse polish notation calculator. -*- c -*- */
/*   Time-stamp: <rpcalc.y (1998/08/18, 9:26:29) by alair at ime.
usp.br)>   */

%{
#define YYSTYPE double
#include <math.h>
%}

%token NUM

%% /* Grammar rules and actions follow */

input:    /* empty */
        | input line
;

line:     '\n'
        | exp '\n'  { printf ("\t%.10g\n", $1); }
;

exp:      NUM                { $$ = $1;          }
        | exp exp '+'        { $$ = $1 + $2;     }
        | exp exp '-'        { $$ = $1 - $2;     }
        | exp exp '*'        { $$ = $1 * $2;     }
        | exp exp '/'        { $$ = $1 / $2;     }
      /* Exponentiation */
        | exp exp '^'        { $$ = pow ($1, $2); }
      /* Unary minus    */
        | exp 'n'            { $$ = -$1;         }
;

%%



/* Lexical analyzer returns a double floating point
   number on the stack and the token NUM, or the ASCII
   character read if not a number.  Skips all blanks
   and tabs, returns 0 for EOF. */

#include <ctype.h>
```

```
yylex ()
{
  int c;

  /* skip white space  */
  while ((c = getchar ()) == ' ' || c == '\t')
    ;
  /* process numbers   */
  if (c == '.' || isdigit (c))
    {
      ungetc (c, stdin);
      scanf ("%lf", &yylval);
      return NUM;
    }
  /* return end-of-file  */
  if (c == EOF)
    return 0;
  /* return single chars */
  return c;
}

main ()
{
  yyparse ();
}

#include <stdio.h>

yyerror (s)  /* Called by yyparse on error */
     char *s;
{
  printf ("%s\n", s);
}
```