```
program FRACTIONS(INPUT, OUTPUT);

+ --     This program performs arithmetic on fractions. All
  -- fractions are computed to the lowest common
  -- denominator. It prints results in fractional form.
  -- A fraction is denoted by a positive integer without a
  -- sign, followed by a slash, and another positive unsigned
  -- integer. These fractions can be combined in general
  -- arithmetic expressions using the conventional operators.
  -- As is the normal convention, multiplication and division
  -- take precedence over addition and subtraction.
  --     The program uses several key variables in performing
  -- its computation. The first is CURRENTVAL. This variable
  -- holds the current value computed for the expression.
  -- Intermediate values are stored on a stack named STACKOFVALS.
  -- This stack is maintained during the course of the
  -- computation, and a pointer to its current depth (STACKPTR)
  -- specifies the number of items currently in the stack.
  -- The maximum stack depth is 100 entries. During the
  -- course of the computation, a number of errors can arise.
  -- These include expressions that are not well-formed
  -- arithmetic expressions, the use of symbolic names in
  -- expressions (which are not allowed), and the use of numbers
  -- with decimal points.
  --     The main program calls three procedures. The first is
  -- GETINPUTEXP. This procedure gets the next expression from
  -- the user of the program. It handles the interaction with
  -- the user. The second procedure is PROCESSEXP. This procedure
  -- takes the string of characters given as input and evaluates
  -- it according to the normal rules of fractional arithmetic.
  -- The third procedure, called PRINTRESULT, prints the result
  -- of the computation. Each of the various procedures can
  -- result in the reporting of one or more errors.
  --     The program can handle successive arithmetic expressions.
  -- These are input one at a time from the user as the user sees
  -- fit. When the user decides that enough expressions have been
  -- entered, the user can terminate the program once and for all
  -- by entering the word STOP in place of an expression. The
  -- entry of this word terminates the program.
  --     The program begins with an introductory message to the
  -- user to enter expressions. Then the user enters an expression
  -- and the result is printed. All results are expressed as
  -- fractions. For example, the result THREE AND ONE-HALF is
  -- printed as 7/2. Errors in expressions input by the user are
  -- flagged as such, and the user must re-enter the entire
  -- expression according to the rules of the program. In these
  -- cases, a special message is given to the user. For example,
  -- the consecutive operators + - are not allowed. +
```

Figure 2.1. Dribbling Comments

```
program FRACTIONS(INPUT, OUTPUT);

+ -- ** PROGRAM TITLE:  FRACTIONS
  --
  -- ** Author:  Cristie L. Jones
  -- ** Date started:  July 10, 1984
  -- ** Date finished: October 5, 1984
  --
  -- ** SUMMARY:
  --         This program takes as input lines of text, each
  --     presenting an arithmetic expression containing
  --     fractions. It computes the fractional result of each
  --     expression.
  --         The program terminates when a line is given
  --     beginning with the word STOP.
  --
  -- ** SAMPLE DIALOGUE:
  --     Computer:  PLEASE ENTER ONE OR MORE LINES EACH
  --     Computer:  REPRESENTING A FRACTIONAL EXPRESSION:
  --     User    :  1/8 + 1/8
  --     Computer:  1/4
  --     NEXT EXPRESSION:
  --     User    :  1/4 + 1/3
  --     Computer:  7/12
  --     Computer:  NEXT EXPRESSION:
  --     User    :  ((3/32)*8
  --     Computer:  ** INVALID INPUT, TRY AGAIN:
  --     User    :  (1/8 + 1/2)*3 - (2/3)/(1/16)
  --     Computer:  -17/8
  --     Computer:  NEXT EXPRESSION:
  --     User    :  STOP
  --
  -- ** INPUT CONVENTIONS:
  --     (a) All values are given as whole, positive integers
  --     (b) The operators +, -, *, and / are allowed
  --     (c) The operators * and / have precedence over + and -;
  --         otherwise evaluation is left to right.
  --     (d) Parentheses indicate grouping
  --     (e) Consecutive operators are not allowed
  --
  -- ** ERRONEOUS INPUTS:
  --     1/-2                needs 1/(-2)
  --     (1/8 + 1/2             needs closing parenthesis
  --     1/2 + +1/2             no successive operators
  --     A/2 + A/2              no symbolic names
  --     1/2 + 0/2              zero not allowed +
```

Figure 2.2. Substantive Comments