```
ATAN(Y/X)
```

should always be changed to

```
ATAN2(Y, X)
```

which correctly handles right-angled triangles instead of causing a division by zero when Y/X is evaluated. Only one change was needed in the function definition to correct all three calculations; we were more likely to get it right. (The program also contains a typographical error:

```
AREA = SQRT(S * (S-BC) * (S-AC) * (S-AB)
```

needs a balancing right parenthesis on the end.)

Fortran's arithmetic statement function is unfortunately restricted to one-line expressions, and is thus of limited usefulness. When the operation to be done is more complex, write a separate subroutine or function. The ease of later comprehending, debugging, and changing the program will more than compensate for any overhead caused by adding the extra modules.

Replace repetitive expressions by calls to a common function.

Another eye-catching repeat appears in

```
      R = 12.
      AL = 24.
      TIME = 0.
      THETA = 0.
      DELTH = 2. * 3.1416 / 100.
      DO 18 I = 1,100
      X = R*(1. - COS(THETA)) + L - L*SQRT(1. - (R*SIN(THETA)/L)**2)
      THETA = THETA + DELTH
      XNEW= R * (1. - COS(THETA)) + L - L*SQRT(1. - (R*SIN(THETA)/L)**2)
      VEL = (XNEW - X) / 0.01
      TIME = TIME + 0.01
   18 WRITE (2,8) TIME, THETA, XNEW, VEL
    8 FORMAT (4F9.2)
      STOP
      END
```

Our first impulse is to define another arithmetic statement function for the gangling expression that appears twice, but closer inspection shows a more fundamental oversight.

The program computes X and its first derivative VEL at each of 100 successive points. Two adjacent values of X must be known to find VEL, so the program dutifully computes both on each iteration, even though one value is already known from the previous iteration. The elaborate expression is computed twice as often as necessary. Worse, it is written twice, which increases the risk that one occurrence will be modified and the other overlooked.

There is also an error: L is used in both expressions where AL is certainly intended. Less serious, but potentially troublesome, is the practice of incrementing a floating point variable many times (see Chapter 6). To keep arithmetic errors