1.4. The average normal human body temperature is usually quoted as 98.6 degrees Fahrenheit, which might be presumed to have been determined by computing the average over a large population and then rounding to three significant digits. In fact, however, 98.6 is simply the Fahrenheit equivalent of 37 degrees Celsius, which is accurate to only two significant digits.

(*a*) What is the maximum relative error in the accepted value, assuming it is accurate to within $\pm 0.05°$ F?

(*b*) What is the maximum relative error in the accepted value, assuming it is accurate to within $\pm 0.5°$ C?

1.10. Consider the expression

$$\frac{1}{1-x} - \frac{1}{1+x},$$

assuming $x \neq \pm 1$.

(*a*) For what range of values of x is it difficult to compute this expression accurately in floating-point arithmetic?

(*b*) Give a rearrangement of the terms such that, for the range of x in part *a*, the computation is more accurate in floating-point arithmetic.

1.5. (*a*) Consider the function

$$f(x) = (e^x - 1)/x.$$

Use l'Hôpital's rule to show that

$$\lim_{x \to 0} f(x) = 1.$$

(*b*) Check this result empirically by writing a program to compute $f(x)$ for $x = 10^{-k}$, $k = 1, \ldots, 15$. Do your results agree with theoretical expectations? Explain why.

(*c*) Perform the experiment in part *b* again, this time using the mathematically equivalent formulation

$$f(x) = (e^x - 1)/\log(e^x),$$

evaluated as indicated, with no simplification. If this works any better, can you explain why?

1.18. Write a program to generate the first n terms in the sequence given by the difference equation:

$$x_{k+1} = 111 - (1130 - 3000/x_{k-1})/x_k,$$

with starting values

$$x_1 = \frac{11}{2} \quad \text{and} \quad x_2 = \frac{61}{11}.$$

Use $n = 10$ if you are working in single precision, $n = 20$ if you are working in double precision. The exact solution is a monotonically increasing sequence converging to 6. Can you explain your results?

1.13. If an amount a is invested at interest rate r compounded n times per year, then the final value f at the end of one year is given by

$$f = a(1 + r/n)^n.$$

This is the familiar formula for *compound interest*. With simple interest, $n = 1$. Typically, compounding is done quarterly, $n = 4$, or perhaps even daily, $n = 365$. Obviously, the more frequent the compounding, the greater the final amount, because more interest is paid on previous interest. But how much difference does this frequency actually make? Write a program that implements the compound interest formula. Test your program using an initial investment of $a = 100$, an interest rate of 5 percent (i.e., $r = 0.05$), and the following values for n: 1, 4, 12, 365. Also experiment with what happens when n becomes very large. Can you find a value such that the final amount does not grow with the frequency of compounding, as it should? (This will be easy if you use single precision, but much harder if you use double precision.)

Implement the compound interest formula in two different ways:

(a) If the programming language you use does not have an operator for exponentiation (e.g., C), then you might implement the compound interest formula using a loop that repeatedly multiplies a by $(1 + r/n)$ for a total of n times. Even if your programming language does have an operator for exponentiation (e.g., Fortran), try implementing the compound interest formula using such a loop and print your results for the input values.

(b) With the functions $\exp(x)$ and $\log(x)$, the compound interest formula can also be written

$$f = a \exp(n \log(1 + r/n)).$$

Implement this formula using the corresponding built-in functions and compare your results with those for the first implementation using the loop, for the same input values.