1) In class we saw an example showing that in decimal floating point arithmetic, the computed value of xmid=(xlower+xupper)/2 is not necessarily between xlower and xupper, which would be a problem for the logic in bisection (in 3 decimal digit arithmetic, try xlower = .997 and xupper = .999). We will show that this is impossible in IEEE arithmetic, which is binary. In other words, we will show that in IEEE arithmetic xmid = \( fl(fl(xlower + xupper)/2) \) is in the interval [xlower,xupper], assuming overflow does not occur when adding xlower and xupper. (Here \( fl(a \text{ op } b) \) means the floating point result of the operation \( a \text{ op } b \).)

**Part 1.** Using the fact that IEEE arithmetic is correctly rounded, show that it is monotonic, that is if \( a, b, c, d \) and \( x \) are IEEE floating point numbers then

\[
\begin{align*}
a &\leq b \text{ and } c \leq d \text{ implies } fl(a + c) \leq fl(b + d) \\
a &\leq b \text{ and } 0 < x \text{ implies } fl(a/x) \leq fl(b/x)
\end{align*}
\]

(Similar facts hold for subtraction and multiplication, but we will not need these here.)

**Part 2.** Show that \( fl(2 \times x) = 2 \times x \) exactly, assuming overflow does not occur.

**Part 3.** Show that \( 2 \times xlower \leq fl(xlower + xupper) \leq 2 \times xupper \).

**Part 4.** Conclude that \( xlower \leq fl(fl(xlower + xupper)/2) \leq xupper \).

**Part 5.** Where does this argument fail for correctly rounded decimal arithmetic?

**Part 6.** What happens if xlower and xupper are adjacent IEEE floating point numbers?
2) Suppose \( x \) is the exact answer to a problem, and \( \hat{x} \) is our approximate answer. In class we defined the absolute error in \( \hat{x} \) as \( |x - \hat{x}| \) and the relative error in \( \hat{x} \) as \( |x - \hat{x}|/|x| \). In this problem we will explore some simple properties of these error measures.

Write the base \( \beta \) expansion of \( x > 0 \) as \( x = .x_1x_2 \cdots x_n \cdot \beta^{e_x} \), and the base \( \beta \) expansion of \( y > 0 \) as \( y = .y_1y_2 \cdots y_n \cdot \beta^{e_y} \). We will say that \( x \) and \( y \) agree to their leading \( d \) base \( \beta \) digits if
\[
|x - y| < \frac{1}{2} \beta^{\max(e_x, e_y) - d}.
\]
For example, .1230 and .1226 agree to 3 decimal digits, as do 1.00 and .996, or .1233 and .1237.

**Part 1.** Suppose you print out \( \hat{x} \) as a base \( \beta \) number. Show that if the relative error \( |x - \hat{x}|/|x| < 1 \), then the leading \( \lfloor \log_\beta |x| \rfloor - 1 \) nonzero base \( \beta \) digits of \( \hat{x} \) are correct, i.e. \( x \) and \( \hat{x} \) agree to that many digits. (\( \lfloor x \rfloor \) is the floor of \( x \), the largest integer less than or equal to \( x \).)

**Part 2.** Suppose you have solved your problem and gotten \( \hat{x} \), and also a bound \( e_{abs} \geq |x - \hat{x}| \) on the absolute error (perhaps using rounding error analysis as described in class). You would like a bound \( e_{rel} \geq |x - \hat{x}|/|x| \) on the relative error. One obvious candidate is \( e_{rel} = e_{abs}/|x| \), but of course you can’t compute this because you don’t know \( x \) (otherwise we wouldn’t need an error bound!). So instead you try \( e_{rel} = e_{abs}/|\hat{x}| \). Show that it is ok to use \( e_{abs}/|\hat{x}| \) instead of \( e_{abs}/|x| \) by showing that
\[
\frac{|x - \hat{x}|}{|x|} \leq \frac{|x - \hat{x}|}{|\hat{x}|} \leq \frac{|x - \hat{x}|}{|x|} \frac{1}{1 - \frac{|x - \hat{x}|}{|x|}}
\]
Conclude that if \( e_{rel} \leq .1 \), then the actual relative error satisfies \( .8e_{rel} \leq |x - \hat{x}|/|x| \leq 1.2e_{rel} \).
3) Let \(1 + r = \prod_{i=1}^{n}(1 + \delta_i)\), where \(|\delta_i| \leq \epsilon < 1\).

**Part 1.** Show that if \(n\epsilon < 1\), then \(|r| \leq n\epsilon/(1 - n\epsilon)\).

**Part 2.** Show that if \(n\epsilon \leq .1\), then \(r \leq 1.2n\epsilon\).

**Part 3.** In IEEE double precision, how big can \(n\) be and satisfy \(n\epsilon \leq .1\)?

**Part 4.** If you compute \(p = \prod_{i=1}^{n}x_i\) in floating point arithmetic, and no over/underflow occurs, and \(n\epsilon \leq .1\), about how many leading decimal digits of the computed value of \(p\) are correct when using IEEE double precision arithmetic with \(n = 10\)? \(n = 100\)? \(n = 1000\)? \(n = 10000\)?
4) Suppose \( x > 0 \). Here are two Matlab algorithms for computing \( e^{-x} \):

**Algorithm 1:** Compute \( e^{-x} \) using a Taylor expansion

\[
\begin{align*}
& s = 1; \quad t = 1; \quad i = 1; \\
& \text{while (abs(t) > eps*abs(s))} \\
& \quad \text{... stop iterating when adding t to s does not change s} \\
& \quad t = -t*x/i; \\
& \quad s = s + t; \\
& \quad i = i + 1; \\
& \text{end} \\
& \text{result1} = s;
\end{align*}
\]

**Algorithm 2:** Compute \( e^{-x} \) as \( 1/e^x \), using a Taylor expansion for \( e^x \)

\[
\begin{align*}
& s = 1; \quad t = 1; \quad i = 1; \\
& \text{while (abs(t) > eps*abs(s))} \\
& \quad \text{... stop iterating when adding t to s does not change s} \\
& \quad t = t*x/i; \\
& \quad s = s + t; \\
& \quad i = i + 1; \\
& \text{end} \\
& \text{result2} = 1/s;
\end{align*}
\]

**Part 1.** Run these two algorithms for \( x = 1:20 \), tabulating the relative errors and number of iterations to converge for each.

**Part 2.** Prove that the relative error of result2 is, as you observe, bounded by \( (3i - 2)\epsilon \), i.e. very accurate. You may assume the error from terminating the Taylor expansion is smaller than round off error, and you may ignore terms proportional to \( \epsilon^2 \). Confirm that \( (3i - 2)\epsilon \) bounds the relative errors in your table above.

**Part 3.** Prove that the relative error of result1 is bounded by \( 3(i - 1)\epsilon e^{2x} \), i.e. it grows quickly with \( x \), so that Algorithm 1 is much less accurate than Algorithm 2. You may make the same assumptions as before. Confirm that \( 3(i - 1)\epsilon e^{2x} \) bounds the relative errors in your table above.

**Part 4.** The computer implementation for \( e^x \) takes the same time for large and small arguments; i.e. it does not use a simple Taylor expansion, which would require more terms for larger arguments. Sketch an algorithm for \( e^x \) that does not take longer for large \( x \). Use the fact that \( e^x = 2^y \) where \( y = x \cdot \log_2 e \), write \( y = y_{\text{int}} + y_{\text{frac}} \) as a sum of an integer and a fraction less than 1, and use the fact that \( 2^y = 2^{y_{\text{int}}} \cdot 2^{y_{\text{frac}}} \) is to be rounded to a floating point number. How many term of a Taylor expansion of \( 2^{y_{\text{frac}}} \) are needed so that the remaining terms contribute less than \( \epsilon \) to the relative error?