1. For each of the following pairs of functions, f(n) and g(n), which of the following are true?

   - $f \sim g$
   - $f = o(g)$
   - $f = O(g)$
   - $g = o(f)$
   - $g = O(f)$
   - $f = \Theta(g)$

   Important: note that if you copy/paste these functions, you will lose the equation formatting! Make sure you are looking at the right functions when you evaluate your answers.

   a. $f(n) = n^2$, $g(n) = 3n + 1$

   b. $f(n) = \frac{3n-7}{n+4}$, $g(n) = 4$

   c. $f(n) = 4^n$, $g(n) = 2^n$

   d. $f(n) = n!$, $g(n) = n^n$

   e. $f(n) = 2^n$, $g(n) = 2^{n/2}$

   f. $f(n) = n^8$, $g(n) = 1.1^n$

2. Two vampires arrive on campus. Each day, every vampire creates a new vampire, and one more vampire arrives on campus. We can write this as a recurrence relation T(n), as such:

$$T(1) = 2$$
$$T(n) = 2T(n-1) + 1$$

where T(n) is the number of vampires on the *n*th day.

Using the guess & verify method, give a closed-form solution for T(n) and a proof that your solution is correct.

Hint: if you are having trouble guessing a closed-form, try the same recurrence with T(1) = 1 and T(1) = 3, and see if those give you a hint.

3. After seeing the lecture about MergeSort, one of your fellow students gets excited, and decides to create an even better version. Instead of splitting the list in half and sorting each half, however, they propose that we split the list in thirds, sort each third, and merge them, as this will surely be faster. They call their new algorithm TriSort.

a. If you had 3 sorted lists, each having $\frac{n}{3}$ elements, what is the worst-case time, in terms of *n*, needed to merge the lists into one sorted list?

b. Using your answer to part a, express the running time for this algorithm as a recurrence of the form T(n) = aT(x) + f(n).

c. Draw the recursion tree of the recurrence you defined in part b.

d. Consider the number of comparisons being done in each layer of the tree. Is it:
   i. $\Theta(1)$ -- always some constant
   ii. $\Theta(n)$ -- always a polynomial where the largest degree is 1
   iii. $\Theta(n^2)$ -- always a polynomial where the largest degree is 2

e. **Optional.** What is the height of the tree?

f. Using either the Master Method, or your answers to d) and e), give a Θ-bound on the running time of this algorithm.

g. Should your fellow student feel triumphant? Or was this a nice try, but better luck next time?