CS361 Homework #1
Due Tuesday, September 12th

1. According to Moore's Law (no relation), computers get twice as fast every two years. Suppose that I can currently solve a problem of size n within one week. What size of problem will I be able to solve in one week with a computer 4 years from now if the running time of my algorithm grows as:
   (a) $f(n) = \Theta(n)$?
   (b) $f(n) = \Theta(n^2)$?
   (c) $f(n) = \Theta(\sqrt{n})$?
   (d) $f(n) = \Theta(2^n)$?
   (e) $f(n) = \Theta(4^n)$?
   (f) $f(n) = \Theta(2^{n/2})$?
   (g) $f(n) = \Theta(\log_2 n)$?
   (h) $f(n) = \Theta(\log_{10} n)$?
2. When we say $f(n) = O(\log n)$, why don't we need to state the base of the logarithm?
3. Is $2^{O(n)}$ the same as $O(2^n)$? Why or why not?
4. There are two ways to represent a graph: an *adjacency matrix* where for every pair of vertices u, v there is a bit that tells you whether u and v are connected, and an *edge list* which gives a list of the edges (u, v). If a graph has N vertices and M edges, how many bits does it take to give the adjacency matrix, and how many bits does it take to give the edge list? Answer both in the *sparse case* where $M = \Theta(N)$, and in the *dense case* where $M = \Theta(N^2)$. Assume that each vertex is represented by a number between 1 and N (how many bits does each such number take?) and give your answers in terms of Θ.
5. For each pair of functions, state whether their relationship is $f = o(g)$, $f = \Theta(g)$, or $f = \omega(g)$:
   (a) $f(n) = \log \sqrt{n}$, $g(n) = \log(n^2)$
   (b) $f(n) = 3^{n/2}$, $g(n) = 2^n$
   (c) $f(n) = 2^n$, $g(n) = n^{\log n}$
   (d) $f(n) = 2^{n+10}$, $g(n) = 2^n$
   (e) $f(n) = 2^{n+\log n}$, $g(n) = 2^n$
   (f) $f(n) = n^n$, $g(n) = n! = n(n-1)(n-2)\cdots 3 \cdot 2 \cdot 1$
6. Substitute a solution of the form $f(n) = A \cdot 4^n + B$ into the recurrence $f(n) = 4f(n-1) + 4$ and the base case $f(0) = 0$ and solve for A and B. Substitute your final solution back in to make sure it works.
7. Substitute a solution of the form $f(n) = r^n$ into the recurrence $f(n) = f(n-1) + 6f(n-2)$ and solve for r. Hint: divide both sides by r^{n-2}. There may be more than one solution; which one matters when n is large?

8. Substitute a solution of the form $f(n) = An$ into the recurrence $f(n) = 2f(n/3) + n$ and solve for A. Substitute your final solution back in to the recurrence to make sure it works.

9. Substitute a solution of the form $f(n) = n^\alpha$ into the recurrence $f(n) = 8f(n/4)$ and solve for α. Substitute your final solution back in to the recurrence to make sure it works.

10. Given the base case $f(0) = 1$ and the recurrence $f(n) = 3f(n-1) + n$, calculate $f(n)$ for the first few values of n, up to $n = 8$ or 9. Find the pattern in these numbers and propose a form for $f(n)$.

    This is somewhat challenging: if you think $f(n)$ grows roughly as fast as some simple function $g(n)$, you might want to check the ratios $f(n)/g(n)$ to figure out the multiplying constant. Then, see if adding some correction term gives you $f(n)$.

    Finally, substitute your final answer back into the recurrence and base case and confirm that it works.

11. In the following program, the global variable `tick` represents the running time. Let $f(n)$ be the total number of `tick`s for calling `calc(n)`. Find the recurrence relation for $f(n)$ and its base cases. Explain which part of the recurrence is the homogeneous term, and which is the driving term. Finally, solve this recurrence and determine $f(n)$ within Θ.

```
calc(int n) {
  if (n <= 0) return;
  else {
    calc(n-1);
    for (int i = 0; i < n; i++)
      for (int j = 0; j < i; j++)
        tick++;
    calc(n-2);
  }
}
```

12. A *quasipolynomial* is a function of the form $f(n) = 2^{\Theta(\log^k n)}$ for some constant $k > 0$ (where $\log^k n$ means $(\log n)^k$).

    (a) Show that a quasipolynomial function with $k > 1$ is ω of any polynomial and o of any exponential.

    (b) Show that if $f(n)$ and $g(n)$ are both quasipolynomial, then so is their composition $f(g(n))$.

13. The research lab of Prof. Flush is well-funded, and they regularly upgrade their equipment. Brilliant Pebble, a graduate student, has to run a rather large simulation. Given that the speed of her computer doubles every two years, if the running time of this simulation exceeds a certain T, she will actually graduate earlier if she waits for the next upgrade to start her program. What is T?