CS361 Homework #5
Due Thursday, December 7th

1. **Cracking the RSA code.** Suppose my public key for the RSA code is $n = 10573$ with an encryption key $e = 5003$. You, the eavesdropper, intercept a message consisting of three four-digit packets:

   2222, 4370, 78

   Your job to is the decrypt this message, using the following method.

   (a) First find the primes p and q such that $pq = n$. (Hint: search for primes which are roughly $\sqrt{n}$).)

   (b) Now find $\phi = (p-1)(q-1)$, which is the number of integers which are mutually prime to n.

   (c) Now, using the extended Euclid's Algorithm, find x and y such that $x\phi + ye = 1$, and thus the inverse $y = e^{-1} \bmod \phi$. This is the decryption exponent d. (Show your work; if y is negative, you'll have to find whatever positive number is equivalent to it mod ϕ. Check your work by making sure that $ed \bmod \phi = 1$.)

   (d) Finally, raise each of the intercepted packets to the dth power mod n and write down the result. Feel free to use any method you like to do this; don't worry about the repeated-squaring thing.

2. For each of these, state whether $f = o(g)$, $\Theta(g)$, or $\omega(g)$. Justify your answer.

   (a) $f(n) = 2^{\sqrt{n}}$, $g(n) = 3^{\sqrt[3]{n}}$

   (b) $f(n) = n^4 3^n$, $g(n) = n^{100}$

   (c) $f(n) = \log(n^2)$, $g(n) = (\log n)^2$

3. Consider the following piece of code:

```
thing(int n) {
  if (n==0) return;
  thing(n-1);
  thing(n-2);
  for (int i=0; i < log(n); i++) print i;
  thing(n-1);
}
```

   Let $f(n)$ be the running time of `thing` on input n; to be specific, let $f(n)$ be the number of times `thing` prints a number. Write a recurrence for $f(n)$, and solve it within Θ.

4. Solve each of these recurrences within Θ and justify your answers.

   (a) $f(n) = 8f(n/2) + n$

   (b) $f(n) = 2f(n-2) + (3/2)^n$

   (c) $f(n) = 2f(n/2) + \sqrt{n}$

   (d) $f(n) = 5f(n/5) + n$

   (e) $f(n) = f(n/2) + 4$

   (f) $f(n) = f(n-1) + f(n-2) + f(n-3) + \cdots + f(1) + f(0)$

5. Solve this recurrence exactly; start with a reasonable guess, and then substitute it back into the recurrence and base case to adjust the constants and make sure it works. (Assume n is a power of 2.)

$$f(n) = 4f(n/2) + 3n^2,\ f(1) = 0$$

6. Suppose I have n people in a room. Assuming that all birthdays are equally likely, what is the average number of triplets of people A, B, C such that their birthdays fall on three consecutive days? How large does n have to be before the probability that there is such a triplet becomes fairly large?

7. If I call `downSift` on a heap containing n items, the worst-case running time is $O(\log n)$. But for most nodes, the running time is only $O(1)$. Explain why, and give a clearer meaning to what "most" and $O(1)$ mean here.

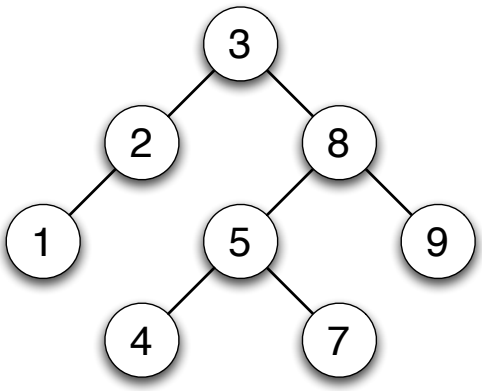

Figure 1: An AVL tree. What happens when we insert 6?

8. Figure 1 shows an AVL tree. What are the balance factors? If I insert a new item with key 6, what will the balance factors be, and which nodes will be "too unbalanced"? What rotations will it do to repair itself, and what will be its final state?

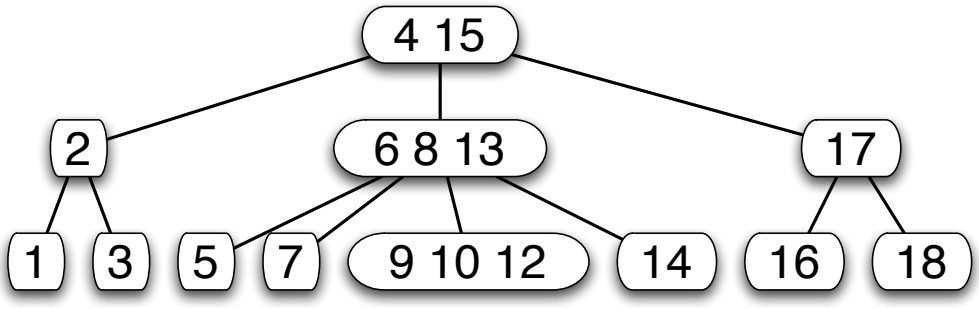

Figure 2: A 2-3-4 tree. What happens when we insert 11?

9. Figure 2 shows a 2-3-4 tree. If I insert a new item with key 11, what will happen, and what will be the final state of the tree? Assume that full nodes get split on the way down the search path.

10. Prove that for any t it is possible to calculate $x^t \bmod p$ with less than $2\log_2 t$ multiplications. For which values of t is this the hardest to do?