Some answers to some student questions: In-class Quiz 2, Spring, 2012.

Questions 8-10 don't make sense to me; I only copied things form the board.

So here they are with some commentary.

#8. "f(n) is O(g(n))" means that there are constants n_0 and c such that f(n) ≤c*g(n) whenever n≥n_0. So to show that n+ 12 is O(n), we need to find constants c and n_0 such that n + 12 ≤ cn whenever n≥n_0. For c, any number > 1 will work. If c = 2, for example, then we need n + 12 ≤ 2n. Solving for n, we get n≥12, so n_0=12 will work. . If c = 3, then we need n + 12 ≤ 3n. Solving for n, we get n≥6, so n_0=6 will work. If we tried c=1, we would get n + 12 ≤ n, which is impossible. We could also have g(n) = n^2, as demonstrated by c=1, n_0 = 12: if n≥12, then n + 12 ≤ n + n = 2n ≤ n*n = n^2.

#9. Again, g(n) can be n, because we know that sin(n) ≤ 1 for all n. So for c=2, n_0=0, n + sin(n) ≤ n + 1 ≤ n + n = 2n.

#10. Again, g(n) = n. Let c=2, n_0=1. If n≥1, then sqrt(n) ≤ n, so n+sqrt(n) ≤ n + n = 2n.

I hope that putting a few more words with the symbols has made this clearer for you. If not, I will be happy to meet with you in person.

What is the difference between the "big" and "little" versions of O and Omega? Think of it like the difference between less-than-or-equal and less-than. "f(n) is O(g(n))" says that f(n) grows no faster than g(n). "f(n) is o(g(n))" says that f(n) grows strictly slower than than g(n). I.e. the f(*n) is O(g(n)) but g(n) is not also O(f(n)). The Omega case is similar, but with faster and slower reversed.

How do you choose the c and n_0? I cannot say very much in general because every function is different. I'll just say that you have to choose them such that the less-than-or-equal-to condition in the definition of big-O is satisfied.Then use what you know about the behavior of the function involved. This is why high school math and calc 1 spend so much time studying the behavior of various functions.

Still don't understand what happened in #14. First, the basis: Consider two positive valued functions f(n) and g(n), and the limit $\lim_{n\to\infty}\frac{f(n)}{g(n)}$.

If the limit is 0, then f(n) is o(g(n)). I.e. f(n) is O(g(n)) but g(n) is not O(f(n)). f(n) grows much slower than g(n).

If the limit is ∞, then f(n) is ω(g(n)). I.e. f(n) is Ω(g(n)) but g(n) is not Ω(f(n)). f(n) grows much faster than g(n).

If the limit is a non-zero constant, then f(n) is Θ(g(n)).

(a) $\lim_{n\to\infty}\frac{n\log n}{n^2}$ = $\lim_{n\to\infty}\frac{n\log n}{n}$= 0 (from example 2 in class). Thus, by the above, n log n is o(n^2).
(b) By the last log formula on p 9 of today's slides(http://www.rose-hulman.edu/class/csse/csse230/201230/Slides/02-MoreIntro-BigOh.pdf), the ratio of these two functions is a constant ($\log_b a$). SO the limit is that same constant. Thus $\log_b n$ is Θ($\log_a n$) and vice-versa
(c) n^a vs a^n. (a > 1) Use l'Hopital's rule (differentiate the numerator and denominator. The numerator is polynomial-type function, so the derivative is a n^{a-1}. Denominator is exponential so the derivative is (log a) a^n. If a-1 is still positive, we do it again to get a(a-1)n^{a-2} and $(\log a)^2$ a^n. No matter how many times we differentiate the denominator, it will always go to infinity as n goes to infinity. For the numerator, after we differentiate enough times, we will get an exponent of 0 (if a is an integer) or negative (if a is not an integer). in either case, the limit of the numerator is 0, If limit of numerator is 0 and limit of denominator is infinite, limit of quotient is 0, so

n^a is $o(a^n)$. There is nothing special about having the same a on both sides. In general a power function is little-o of an exponential function, as long as the base of the exponential is at least 1.

Proving something by induction. In general you need to directly show that the base case is true. Then show that for any integer k that is at least as large as the base case, if we assume that the property is true for that k, it also must be true for the next integer, k+1. All of the rest is in the details of how to get from the k case to the k+1 case and that is different for every problem. We will see several more examples.

I just need to go through big-Oh notation myself. Everyone should do that! If you still have trouble, I will be happy to meet with you to talk about it.