Ten Lessons I Learned as a Programmer from CSE 351

1. When working with extraordinarily large or small numbers, I will be careful to take into account the precision limits for integers and floats; I will store values in log form when I suspect I will run into overflow or underflow issues.
2. I will watch out for bugs due to implicit casting from mixing signed and unsigned values; I will double-check the declared type of parameters before calling a function that takes a signed/unsigned value.
3. When writing C programs, I will not create a variable on the stack and then try to return it to the caller function. I now know why that doesn't work. I will use malloc() and free() instead.
4. I will use compiler optimizations that help turn my code into more efficient versions of the assembly. I will use powers of 2 when reasonable so the compiler can make use of leaq instead of imulq.
5. I will forgive my computer when I have a lot of applications running at once and the computer slows down. I will know that it is doing significant context switching and I will try to close out of a few things if possible. On the programming side, I will keep my fork() structure as simple as possible (though no simpler).
6. I will never use gets() or strcpy()! Knowing the power of buffer overflows helps me stay away from functions that introduce security vulnerabilities.
7. I will use good spatial and temporal locality in my code, regardless of whether I'm using C or Java. For instance, I will access arrays with the fastest-changing index on the right.
8. When I need to optimize around a bizarre array access pattern, I will try to craft a blocked version of my algorithm to make better use of the cache.
9. I now know how memory leaks happen and I will work to prevent them in my code. I will strive to always free exactly once that which I have malloc-ed, and I will run valgrind to try to catch mistakes.
10. When I need extra speed in my program, I will opt for C as opposed to Java. When I need extra speed in my Java program, I will opt for primitive types instead of object references.