Lab 3

1. $10000000000n^2$ vs n^3

n^3 is asymptotically greater bc as n goes to infinity, n^3 grows faster than n^2 meaning that n^3 is asymptotically greater. In this case since n^3 is a higher degree then n^2, we know that n^3 will eventually grow faster than n^2, even though n^2 is being multiplied by a really long constant.

2. $n^2 \log n$ vs $n(\log n)^{10}$

For this comparison, we can first compare like terms. Since n^2 is a higher degree than n, we know it grows faster. Next we know $\log n$ grows slower than $(\log n)^{10}$. The two parts that grow fast are n^2 from the first equation and $(\log n)^{10}$ from the second equation. For that reason, we can compare the 2 and see n^2 grows much faster, so the asymptotically greater one will be $n^2 \log n$.

3. $n^{\log n}$ vs $2^{\sqrt{n}}$

$n^{\log n}$ should be bigger because although for smaller values of n, $2^{\sqrt{n}}$ will be bigger but because $n^{\log n}$ has an n is the bottom and the exponent, for bigger numbers it will definitely grow faster than $2^{\sqrt{n}}$. So we know that $n^{\log n}$ is asymptotically greater.

4. 2^n vs 2^{2n}

Although 2^{2n} will always grow faster because it has $2n$ in the exponent and the base is the same, I believe that the asymptotic notations are the exact same because in asymptotic notation we are not worried about constants so we should get a tight bound of $\Theta(2^n)$ for both functions, meaning that they are asymptotically the same.

Lab 3 continued

2. isPrime(n): 1 time

```
for (i=2, i*i <= n; i++) {
    if n %% i == 0 {
        return false
    }
}
return true
```

Best case: If n is divisible by 2, then the first while loop iteration will return false because n %% 2 will be 0, meaning that the function will return false. looking at the lower bound the for loop will at least run once if a valid n is given so we have $\Omega(1)$ lower bound. The maximum time that it will run in the best case is also 1 because best case means it is divisible by 2, so the upper bound will be $O(1)$. Since $\Omega(1) = O(1)$ we get $\Theta(1)$ for the tight bound of the best case.

worst case: Since i keeps multiplying itself to make sure that multiplying itself doesn't go over n. That means that we must go through $\sqrt{n}$ elements to check all possible pairs of division. For a number only divisible by itself and 1, the minimum time run will be going through all necessary elements until the for loop fails which is $\Omega(\sqrt{n})$ time and the max is $O(\sqrt{n})$ time. Since $\Omega(\sqrt{n}) = O(\sqrt{n})$, we get $\Theta(\sqrt{n})$

Average case: on average we would get something more than constant time, which would be the lower bound $\Omega(1)$, and something less than $\sqrt{n}$ time which is the upper bound, $O(\sqrt{n})$. Since $\Omega(1) \neq O(\sqrt{n})$, we shouldn't have a tight bound, but we then would have to go with $O(\sqrt{n})$ for the closest approximation of the average time case complexity.