Problem 1:

Let's compare some basic math functions to refresh our memory. For each of the following, just write which function is *asymptotically greater* (So, you should be thinking about your asymptotic notations!). Show your reasoning for the same.

1. $10000000000n^2$ vs n^3 in the long run n^3 will be greater because "10000000000" is a finite number.
2. $n^2 \log(n)$ vs $n(\log(n))^{10}$ the $n^2\log(n)$ is greater than $n(\log(n))^{10}$ because it is a quadratic function and the $n(\log(n))^{10}$ is linear.
3. $N^{\log n}$ vs 2^√n the 2^√n function will be greater than $N^{\log n}$ for sufficiently large values of n because the logarithmic exponent causes the growth to be sub-exponential.
4. 2^n vs 2^{2n} The 2^2n is greater because it is double the exponential growth.

Problem 2:

Now let's examine some [pseudo]code and apply asymptotic notation to it.

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

What is the

1. Best Case: O(1)
2. Worst Case: O($\sqrt{n}$)
3. Average Case: O($\sqrt{n}$)

Time complexity for the above function? Time complexity is O($\sqrt{n}$).