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: n^3 is asymptotically greater than n^2. No matter how large the coefficient for n^2, as long as it is a constant n^3 will be asymptotically greater.
2. $n^2 \log(n)$ vs $n(\log(n))^{10}$: n(logn)^10 will be asymptotically greater because it grows faster
3. $n^{\log n}$ vs 2^n: n^logn is asymptotically greater because between base n and base 2, base n will grow faster than the constant 2.
4. 2^n vs 2^{2n}: asymptotically these two should be the same because they have the same base of 2 and they are to the power of n. The only difference is the constant in front of the second n, however asymptotically this would not make a difference.

Problem 2:

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

```
isPrime(n):
 for(i = 2, i*i <= n; i++) { //i^2 = n—> i = sqrt(n)
   if(n % i == 0) { //O(1)
     return false //O(1)
   }
 return true // O(1)
```

What is the

1. Best Case: O(1) **if n is 2 then n mod 2 would be 0 and return false in the first iteration.
2. Worst Case: O(sqrt(n))
3. Average Case:O(sqrt(n))

Time complexity for the above function? O(sqrt(n)) because on average the function has to loop through the entire for loop which increases by one and stops after i reaches a value of square root of n.