Written Assignment 30 pts

1. Big-O Notation (18pts)

Find tightest $O(f(n))$ for each of the following functions: The tightest big-O bound is the narrowest upper bound within the big-O category.

Example: n = O(n^2), n = O(n^4).. and so on, but the tightest bound would be n = O(n)

Example: $f(n) = 5n^2 + 2n + 1 = O(n^2)$

a. $f(n) = 3n$ $O(n)$ ← 3 is constant

b. $f(n) = \frac{log(n)}{n^2}$ $O\left(\frac{\log(n)}{n^2}\right)$ ← tighter than $O(1)$

c. $f(n) = n \times logn$ $O(n \times \log(n))$

d. $f(n) = n + \frac{n}{2} + \frac{n}{4} + ... + \frac{n}{2^n}$ $O(2n)$

e. $f(n) = (log(n))^n + n^4$ $O(\log(n)^n)$

f. $f(n) = \frac{n! + n^n}{3n}$ $O(n^{n-1})$

$\cancel{\frac{n!}{3n}} + \frac{n^n}{3n} = \frac{1}{3} + \frac{n^n}{3n} = \frac{1\cancel{n} + n^{n-1}}{3\cancel{n}}$ (1/3 constant)

True/False.

Show you work by using the definition of big-O and finding values for c and N.

Reminder $f(n)$ is $O(g(n))$ — if a positive real number c and positive integer N exist such that $f(n) \leq c \times g(n)$ for all $n \geq N$

g. $2^{n-1} = O(n)$ False

h. $n(logn)^3 = O(n^{4/3})$ True

i. $\frac{n^4+1}{n^2} = O(n)$ False

$n(\log(n))^3 = O(n^{4/3})$

$\frac{n(\log(n))^3}{n} \leq \frac{Cn^{4/3}}{n}$

$\cancel{\sqrt[3]{\log(n)^3}} \leq \sqrt[3]{Cn^{4/3}}$ $(n^{3/3})(n^{1/3})$

$\log(n) \leq Cn^{1/9}$

$\cancel{n} \leq C\frac{10^{n^{1/9}}}{n}$

$1 \leq \frac{n}{10^{n^{1/9}}}$

2. Algorithmic Analysis (12pts)

Given the following code, analyze and give the tightest big-Θ bound. Show how you came to your answer by indicating what the big-Θ is for each line.

a.
```
public static int sum1() {
        int sum = 0;
        for(int i = 0; i < n; i++) {
             if(sum < n) {
                  for(int j = 0; j < n; j++) {
                       sum++;
                  }
             }
        }
        return sum;
   }
```

← runs "n" times

$\underbrace{j < n}$ ↳ once

$n + n = 2n$

$\Theta(n)$

b.
```
public static int sum2() {
        int sum = 0;
        for(int i = n; i > 1; i = i/3) {
             sum = sum + 2;
        }
        return sum;
   }
```

$\overbrace{i = i/3}^{\text{log function}}$

$\Theta(\log(n))$

c.
```
public static int sum3() {
        int sum = 0;
        for(int i = 0; i < n; i++) {
             for(int j = 0; j < n; j++) {
                  if(i < j) {
                       for(int k = i; k < j; k++) {
                            sum++;
                       }
                  }
             }
        }
        return sum;
   }
```

→ $1+2+3+\ldots+(n-1)$

→ $1+2+3+\ldots+n = \frac{(n-1)(n)}{2}$

(limit ish)

↳ Runs N times

$\frac{(n-1)(n)}{2} \to n^2$

$n^2 * n = n^3$

$\Theta(n^3)$