Name: Breona Pizzuta Date: 9/23/24

I pledge my honor that I have abided by the Stevens Honor System.

Point values are assigned for each question. Points earned: ____ / 100, = ____ %

1. Find a tight upper bound for $f(n) = n^4 + 10n^2 + 5$. Write your answer here: $O(n^4)$ (4 points)

   Prove your answer by giving values for the constants c and n_0. Choose the smallest integer value possible for c. (4 points)

$$0 \leq n^4 + 10n^2 + 5 \leq cn^4$$

$n^4 + 10n^2 + 5 \leq 2n^4 \ \ when\, n \geq 4$

$c = 2$ and $n_0 = 4$

2. Find an asymptotically tight bound for $f(n) = 3n^3 - 2n$. Write your answer here: $\theta(n^3)$ (4 points)

   Prove your answer by giving values for the constants c_1, c_2, and n_0. Choose the tightest integer values possible for c_1 and c_2. (6 points)

We need to show that $c_1 n^3 \leq 3n^3 - 2n \leq c_2 n^3$

For the lower bound we can say c_1= 2 because $2n^3 \leq 3n^3 - 2n$ when $n \geq 2$

For the upper bound we can say c_2= 3 because $3n^3 - 2n \leq 3n^3$ when $n \geq 1$

Therefore: $c_1 = 2$; $c_2 = 3$; $n_0 = 2$

3. Is $3n - 4 \in \Omega(n^2)$? Circle your answer: yes / no. (2 points)

   If yes, prove your answer by giving values for the constants c and n_0. Choose the smallest integer value possible for c. If no, derive a contradiction. (4 points)

$$\lim_{n\to\infty} \left(\frac{3n-4}{n^2}\right) = \lim_{n\to\infty} \left(\frac{3}{n} - \frac{4}{n^2}\right) = 0$$

Or to explain:
So let's assume that there exists positive constants c and n such that$_0$
$0 \leq cn^2 \leq 3n - 4$ ($\forall n \geq n_0$)
We also have $3n-4 \leq 3n$($n \geq 1$)

So by combining the two we must have:
$cn^2 \leq 3n$ ($\forall n \geq \max(n_o, 1)$)
$cn \leq 3$ ($\forall n \geq \max(n_o, 1)$)
$n \leq 3/c$ ($\forall n \geq \max(n_o, 1)$)
Since n can grow to infinity, it is impossible to find a positive constant c such that n is bounded above by the constant 3/c. Therefore the c we need to find does not exist. Therefore
$3n - 4 \notin \Omega(n^2)$

4. Write the following asymptotic efficiency classes in **increasing** order of magnitude.
$O(n^2)$, $O(2^n)$, $O(1)$, $O(n \lg n)$, $O(n)$, $O(n!)$, $O(n^3)$, $O(\lg n)$, $O(n^n)$, $O(n^2 \lg n)$ (2 points each)

   $O(1)$, $O(\lg n)$, $O(n)$, $O(\mathrm{n} \lg n)$, $O(n^2)$, $O(n^2 \lg n)$, $O(n^3)$, $O(2^n)$, $O(n!)$, $O(n^n)$

5. Determine the largest size *n* of a problem that can be solved in time *t*, assuming that the algorithm takes *f(n)* milliseconds. Write your answer for *n* as an integer. (2 points each)

   a. *f(n)* = n, *t* = 1 second 1000

   b. *f(n)* = $n \lg n$, *t* = 1 hour 204094

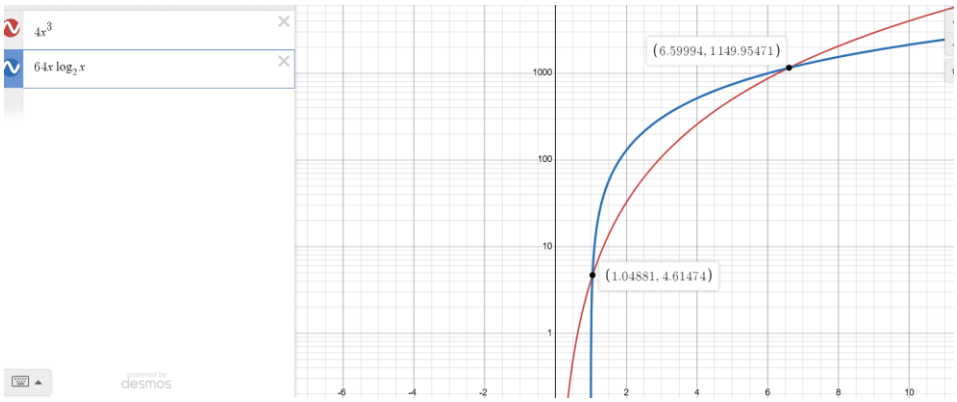
   c. *f(n)* = n^2, *t* = 1 hour 1897

   d. *f(n)* = n^3, *t* = 1 day 442

   e. *f(n)* = $n!$, *t* = 1 minute 8

6. Suppose we are comparing two sorting algorithms and that for all inputs of size n the first algorithm runs in $4n^3$ seconds, while the second algorithm runs in $64n \lg(n)$ seconds. For which integer values of n does the first algorithm beat the second algorithm?
   The first algorithm beat the second algorithm when n= [2,6] (4 points)

   Explain in detail how you got your answer or paste code that solves the problem (2 point):
   When plotting the two algorithms on a graph, we observe that in the range n=2 to n=6, the curve for the first algorithm ($4n^3$) lies below the curve for the second algorithm ($64n \lg(n)$). This indicates that the first algorithm is faster for these input sizes. The intersection point at n=6 marks the point where both algorithms take the same time. Beyond n=6, the second algorithm becomes faster as the growth of the first algorithm begins to dominate. This can be shown in the provided graph.



Give the complexity of the following methods. Choose the most appropriate notation from among O, Θ, and Ω. (8 points each)

```
int function1(int n) {
    int count = 0;
    for (int i = n / 2; i <= n; i++) {              //n/2 times
        for (int j = 1; j <= n; j *= 2) {          //log  n times
            count++;
        }
    }
    return count;
}
```

Answer: $\Theta(n \log_2 n)$

```
int function2(int n) {
    int count = 0;
    for (int i = 1; i * i * i <= n; i++) {  //cubic iteration n^1/3
        count++;
    }
    return count;
}
```

Answer: $\Theta(n^{1/3})$

```
int function3(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {                // n times
        for (int j = 1; j <= n; j++) {            // n times
            for (int k = 1; k <= n; k++) {        // n times
                count++;
            }
        }
    }
    return count;
}
```

Answer: $\Theta(n^3)$

```
int function4(int n) {
    int count = 0;
```

```
    for (int i = 1; i <= n; i++) {        // n times
        for (int j = 1; j <= n; j++) {     // n times
            count++;
            break;                    //break !!!
        }
    }
    return count;
}
```

Answer: θ (n)

```
int function5(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {      //n times
        count++;
    }
    for (int j = 1; j <= n; j++) {      //n times
        count++;
    }
    return count;
}
```

Answer: θ(n)