Name: ______________________________ Date: __________________

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

1. Find an 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 integral value possible for c. (4 points)

$$0 \leq n^4 + 10n^2 + 5 \leq c \cdot n^4 \; \forall n \geq 4$$
$$\begin{cases} c = 2 \\ n_0 = 4 \end{cases}$$

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 integral values possible for c_1 and c_2. (6 points)

$$0 \leq c_1 \cdot n^3 \leq 3n^3 - 2n \leq c_2 \cdot n^3 \; \forall n \geq 1$$
$$\begin{cases} c_1 = 2 \\ c_2 = 3 \\ n_0 = 2 \end{cases}$$

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 integral value possible for c. If no, derive a contradiction. (4 points)

$$c \cdot n^2 \leq 3n - 4 \leq 3n$$
$$c \cdot n^2 \leq 3n$$
$$c \cdot n \leq 3$$
$$n \leq \frac{3}{c}$$
$$\therefore \text{a contradiction, holds for } n \leq \text{a constant}$$
$$\text{Other valid answers are possible, as long as they show } n \leq \text{a constant.}$$

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)

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

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 point each)

   a. *f(n)* = n, *t* = 1 second ___1000___ $n = 1s \cdot \frac{1000ms}{s} = 1000ms = 1000 \text{ operations}$

   b. *f(n)* = $n \lg n$, *t* = 1 hour ___204,094___ $n \lg n \leq 1hr \cdot \frac{60min}{hr} \cdot \frac{60s}{min} \cdot \frac{1000ms}{s} = 3{,}600{,}000$

(solve numerically)

c. *f(n)* = n^2, *t* = 1 hour ___1,897___ $n^2 \leq 3{,}600{,}000 \Rightarrow n = 1897$

d. *f(n)* = n^3, *t* = 1 day ___442___ $n^3 \leq 86{,}400{,}000 \Rightarrow n = 442$

e. *f(n)* = $n!$, *t* = 1 minute ___8___ $n! \leq 60{,}000 \Rightarrow n = 8$

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 integral values of n does the first algorithm beat the second algorithm? ___$2 \leq n \leq 6$___ (4 points) Explain how you got your answer or paste code that solves the problem (2 points):
Solved numerically with computer program. See below.

```
def solve():
    lower = 0
    upper = 0
    n = 1
    while True:
        left = 4 * (n ** 3)
        right = 64 * n * math.log(n, 2)
        if lower == 0:
            if left <= right:
                lower = n
        elif left > right:
            upper = n - 1
            break
        n += 1
    print("%d <= n <= %d" % (lower, upper))
```

Output: `2 <= n <= 6`

7. 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++) {
        for (int j = 1; j <= n; j *= 2) {
            count++;
        }
    }
    return count;
}
```

Answer: $\theta(n \lg n)$

```
int function2(int n) {
    int count = 0;
    for (int i = 1; i * i * i <= n; i++) {
        count++;
    }
    return count;
}
```

Answer: $\theta(\sqrt[3]{n})$

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

Answer: $\theta(n^3)$

```
int function4(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            count++;
            break;
        }
    }
    return count;
}
```

Answer: $\theta(n)$

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

Answer: $\theta(n)$