Name: Matthew McCaughan Date: 9/28/2022

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 an upper bound for $f(n) = n^4 + 10n^2 + 5$. Write your answer here: **$2n^4$** (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)

   $n^4 + 10n^2 + 5 \leq 2n^4$ $(\forall n \geq 4)$
   $c = 2$
   $n_0 = 4$

2. Find an asymptotically tight bound for $f(n) = 3n^3 - 2n$. Write your answer here: **$3n^3 \leq f(n) \leq 4n^3$** (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)
   $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$

   $0 \leq 3n^3 \leq 3n^3-2n \leq 4n^3$ $(\forall n \geq 1)$
   $c_1 = 3$
   $c_2 = 4$
   $n_0 = 1$

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)
   We need to find positive constants c and $n_0$ such that:
   $0 \leq cn^2 \leq 3n-4$ $(\forall n \geq n_0)$
   $3n-4 \leq 3n$ $(\forall n \geq 1)$

   Therefore:
   $cn^2 \leq 3n$ $(\forall n \geq \max(n_0, 1))$
   $cn^2-3n \leq 0$ $(\forall n \geq \max(n_0, 1))$
   $n(cn-3) \leq 0$ $(\forall n \geq \max(n_0, 1))$
   $cn-3 \leq 0$
   $n \leq 3/c$

   n cannot be bounded by any constant c as it grows, so there is no c that exists. Therefore **$3n-4 \notin n^2$**

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

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

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 n = 1000 milliseconds 1000

   b. *f(n)* = $nlg\ n$ , *t* = 1 hour n*lg(n) = 3600000 204094 (computed in python file on canvas)

   c. *f(n)* = $n^2$, *t* = 1 hour $n^2$ = 3600000 1897

   d. *f(n)* = $n^3$, *t* = 1 day $n^3$ = 86400000 442

   e. *f(n)* = $n!$, *t* = 1 minute n! = 60000 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? (n ≤ 2) (4 points)
   Explain how you got your answer or paste code that solves the problem (2 point):
   Since $O(n^3)$ increases in magnitude much greater than $O(n\ lg\ n)$, the range for which $4n^3$ will beat $64nlg(n)$ will be small. When **n = 2,** $(4(2)^3)$ ≤ (64(2)*log(2)), and when **n = 3,** $(4(3)^3)$ ≥ (64(3)*log(3)). This means that 2 is the threshold for which the first algorithm will beat the second algorithm.

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

Answer: **Θ(n*log(n))**

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

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

```
int function3(int n) {
    int count = 0;                                        // constant
    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++;                                  // constant
            }                                             // n*n*n = n^3
        }
    }
    return count;
}
```

Answer: $\Theta(n^3)$

```
int function4(int n) {
    int count = 0;                          //constant
    for (int i = 1; i <= n; i++) {          // n times
        for (int j = 1; j <= n; j++) {      // n times
            count++;                        //constant
            break;                          //c_1*c_2*n*n = n^2
        }
    }
    return count;
}
```

Answer: $\Theta(n^2)$

```
int function5(int n) {
    int count = 0;                          // constant
    for (int i = 1; i <= n; i++) {          // computes n times
        count++;                            // constant
    }
    for (int j = 1; j <= n; j++) {          // computes n times
        count++;                            // constant
    }
    return count;                           // (c_1+c_2)n + (c_1)n = n
}
```

Answer: $\Theta(n)$