Name: Liam Brew Date: 02.10.2020

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

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)

$$0 \leq n^4 + 10n^2 + 5 \leq cn^4 \rightarrow n^4 + 10n^2 + 5 \leq 2n^4 \; \forall n \geq 4$$

c = 2, n_0 = 4

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

   Upper Bound: $3n^3 - 2n \leq 3n^3 \; \forall n \geq 1$
   Lower Bound: $3n^3 - 2n \leq 2n^3 \; \forall n \geq n$

   Therefore c_1 = 2, c_2 = 3 and 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 integral value possible for c. If no, derive a contradiction. (4 points)

$$\lim_{n\to\infty} (\frac{3n-4}{n^2})^1 = \lim_{n\to\infty} \left(\frac{3}{n} - \frac{4}{n^2}\right)^1 = 0$$

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(lgn), O(n), O(nlgn), O(n^2), O(n^2lgn), O(2^n), O(n^3), 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. *n* must be 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 integral values of n does the first algorithm beat the second algorithm? n = [2, 6] (4 points)
Explain how you got your answer or paste code that solves the problem (2 point):

```
# Determines the integral values for which an algorithm of runtime 4n^3 seconds b
eats an algorithm of runtime 64nlgn seconds.
def functionA(n):
    # 4n^3
    counter = 0
    for i in range(0, 4):
        for j in range(0, n):
            for k in range(0, n):
                for l in range(0, n):
                    counter += 1
                    l += 1
                k += 1
            j += 1
        i += 1
    return counter
def functionB(n):
    # 64nlgn
    counter = 0
    for i in range(0, 64):
        for j in range(0, n):
            k = 1
            while k < n:
                counter += 1
                k *= 2
            j += 1
        i += 1
    return counter
if __name__ == "__main__":
    n = 0
    for n in range(0, 100):
        if functionA(n) < functionB(n):
    print(n)
```

```
D:\shared>C:/Users/liamb/AppData/Local/Programs/Python/Python38-32/python.exe "d:/shared/Programming Assignments/Homework
/algorithm analysis/hw1b#6.py"
2
3
4
5
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: $\underline{\Theta(n\lg n)}$

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

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

```
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: $\underline{\Theta(n^3)}$

```
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: $\underline{\Theta(n)}$

```
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: $\underline{\Theta(n)}$