Name: _____Kevin Wittlinger_____ Date: _____9/27/23_____

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

"I pledge my honor that I have abided by the stevens honor system"

- Find a tight upper bound for $f(n)$ = n4 +10n2 +5. Write your answer here: __O(n^4)____ (4 points)

  Prove your answer by giving values for the constants c and n0. Choose the smallest integer value possible for c. (4 points)

  c=2

  n0=4

- Find an asymptotically tight bound for $f(n)$ = 3n3 —2n.Write your answer here: Ω(n^3)____ ((4 points)

  Prove your answer by giving values for the constants c1, c2, and n0. Choose the tightest integer values possible for c1 and c2. (6 points)

  c1= 2

  c2= 3

  n0= 2

- Is 3n — 4 ϵ Ω(n2)? Circle your answer: yes / no. (2 points)

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

  0 <= cn^2 <= 3n-4 ($\forall n \geq n_0$)

  3n-4 <= 3n-4n ($\forall n \geq 1$) = -1n

  cn^2 <= -1n ($\forall n \geq \max(n_0, 1)$)

  cn ≤ -1($\forall n \geq \max(n_0, 1)$)

  n ≤ -1/c ($\forall n \geq \max(n_0, 1)$)

n is able to grow to infinity meaning it is impossible to find a positive constant c such that n is bounded by the constant of 15/c. This is impossible because n grows with no bound while 15/c remains a finate number. Therefore, the c we need doesn't exist and f(n)= $3n - 4 \notin \Omega(n2)$?

- Write the following asymptotic efficiency classes in **increasing** order of magnitude.

  $O(n2)$, $O(2n)$, $O(1)$, $O(n\lg n)$, $O(n)$, $O(n!)$, $O(n3)$, $O(\lg n)$, $O(nn)$, $O(n2\lg n)$ (2 points each)

  _$O(1)$, _$O(\lg n)$, _ $O(n)$, __$O(n\lg n)$_,_$O(n\text{^}2)$, __$O(n\text{^}2\lg n)$ _,_$O(n\text{^}3)$, _$O(2\text{^}n)$, _ $O(n!)$,

- 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) = nlg n, t = 1 hour _204094_____

  c. f(n) = n2, t = 1 hour ____1897_____

  d. f(n) = n3, t = 1 day ____441_____

  e. f(n) = n!, t = 1 minute ___8_____

- Suppose we are comparing two sorting algorithms and that for all inputs of size the first algorithm runs in seconds, while the second algorithm runs in seconds. For which integer values of does the first algorithm beat the second algorithm? ____n <= 6_____ (4 points)

  Explain in detail how you got your answer or paste code that solves the problem (2 point):
  Set 4n3 < 64nlogn then simplified to n2 < 16 logn. You then go through iterations of n plugging in each one till you reach a value that has a greater value on first algorithm than the second algorithm. Below is a chart made to show each iteration:

n	n^2	16lg(n)
1	1	0
2	4	16
3	9	25

4	16	32
5	25	37
6	36	41
7	49	45

In this case that is at n=7 where the n^2 reaches 49 and right side is aproximatly 45 meaning that the second algorithm is now faster.

- 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: ___Θ(nlogn)_____

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

Answer: __Θ(^3sqrt(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: ____Θ(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: ___Θ(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: _____Θ(n)____