­Name: \_\_\_\_\_Joshua Schmidt\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_9/16/2019\_\_\_\_\_

*I pledge my honor that I have abided by the Stevens Honor System.* - Joshua Schmidt

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

1. Find an upper bound for . Write your answer here: \_\_\_\_\_\_\_ (4 points)

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

1. Find an asymptotically tight bound for . Write your answer here: \_\_ (4 points)

Prove your answer by giving values for the constants , , and . Choose the tightest integral values possible for and . (6 points)

answer:

1. Is Circle your answer: yes / **no**. (2 points)

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

contradiction:

if , 3n-4 should be greater than n2, but the expression above shows it doesn’t work for n=1.

The function has a value lower than Omega, which should not be possible.

Generalized:

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

, , , , , , , , (2 points each)

\_\_,\_\_,\_\_,\_\_, \_, \_, \_, \_\_,\_\_, \_\_

1. 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)
2. *f(n)* = , *t* = 1 second \_\_1000\_\_
3. *f(n)* = , *t* = 1 hour \_\_204094\_\_\_
4. *f(n)* = , *t* = 1 hour \_\_1897\_\_\_
5. *f(n)* = , *t* = 1 day \_\_442\_\_\_\_
6. *f(n)* = , *t* = 1 minute \_\_8\_\_\_\_\_\_
7. 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 integral values of does the first algorithm beat the second algorithm? \_\_\_\_\_\_\_\_\_\_ (4 points)

Explain how you got your answer or paste code that solves the problem (2 point):

\_\_I tried different values of n, starting at n=1 (undefined), until the runtime for the first algorithm became larger than the second algorithm. This occurred when n=7. Then I concluded that n must be between 2 and 6 in order for the statement to work. I essentially found the intersection between the two functions.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Give the complexity of the following methods. Choose the most appropriate notation from among , , and . (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: \_\_\_\_\_

**int** function2(**int** n) {

**int** count = 0;

**for** (**int** i = 1; i \* i \* i <= n; i++) {

count++;

}

**return** count;

}

Answer: \_\_\_\_\_

**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: \_\_\_\_\_

**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: \_\_\_\_

**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: \_\_\_\_