Name: \_\_**Katie Prescott**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_**2/13/16**\_\_\_\_\_\_\_\_

**I pledge my honor that I have abided by the Stevens honor system.**

Point values are assigned for each question. Points earned: \_\_\_\_ / 42, = \_\_\_\_ %

1. Use the definitions of , Θ, and Ω to determine whether the following assertions are true or false. If true, give values for the appropriate constants. If false, explain the contradiction. (3 pts. each)
   1. \_**True: ,** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. \_**True: ,** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. \_\_**False: , can’t have n less than or equal to a constant.** \_\_\_\_\_\_\_\_
   4. \_\_**False: , can’t have n less than or equal to a constant.** \_\_\_\_\_\_\_\_
2. Write the following asymptotic efficiency classes in **increasing** order of magnitude.

(1 pt. 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. (1 pt. each)
   1. \_\_\_\_\_\_\_\_\_
   2. \_\_\_\_\_\_\_\_\_
   3. \_\_\_\_\_\_\_\_\_
   4. \_\_\_\_\_\_\_\_\_
   5. \_\_\_\_\_\_\_\_\_
2. (3 pts.) 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? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Explain how you got your answer: **Based on the graphs, you can tell where the first function beats the second function by checking where they intersect, (where *)* which is and at that point. The first function beats the second function where because that is where**  **is less than** **.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Give the complexity of the following methods. Choose the most appropriate notation from among . (3 pts. 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 <= 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: \_\_\_ \_\_\_\_\_\_