Name: _____

# CS443: Analysis of Algorithms Midterm, Sp 03

| Score: | 1. | (max 10) | 5. | (max 12) |
|---|---|---|---|---|
| | 2. | (max 12) | 6. | (max 12) |
| | 3. | (max 12) | 7. | (max 12) |
| | 4. | (max 18) | 8. | (max 12) |
| | Total: | | (max 100) | |

*This exam is closed book. Calculators are not allowed.*

1. (10 pts) State the definition of $f(n) = \Omega(g(n))$.

2. (12 pts) Use the definition you gave in the previous problem to prove or disprove the claim: $h_1(n) + h_2(n) = \Omega(max(h_1(n) + h_2(n))$. Carefully explain all of your steps.

3. (12 pts) Order the following functions from smallest to largest based on their Big-Oh complexity. Be sure to identify which functions are the same (that is, which are Big-Theta of one another). Assume logs are base 2.

   $n\log(n)$ $\quad 100^2$ $\quad n!$ $\quad n^3$ $\quad 2^{2^n}$ $\quad \sqrt{n}$ $\quad 2^{2log(n)}$ $\quad \log^2(n)$

4. (3 pts each, 18 pts total) Circle true or false:

   (a) $f(n) = \Omega(f(n))$ true or false

   (b) $f(n) = \Omega(g(n)) \Rightarrow g(n) = O(f(n))$ true or false

   (c) $f(n) = O(g(n)) \Rightarrow f(n) = o(g(n))$ true or false

   (d) $f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$ true or false

   (e) $f(n) = O(g(n)) \Rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ true or false

   (f) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) = \omega(g(n))$ true or false

5. (12 pts) Use induction to prove the following. Be sure to explain what you are doing in each step.

$$\sum_{i=0}^{n} x^i = \frac{1 - x^{n+1}}{1 - x}$$

6. (12 pts) Suppose you are given an array containing $n$ numbers. You call *build-heap* on it so that the result is a heap. The largest item will be in the array at index 0. Give *all* the possible places (i.e. array indices) where it would be possible to find the smallest item. Be specific and explain your answer. Show some examples in tree and/or array form to help explain your answer.

7. (12 pts) Describe at least two different collision detection methods for hash tables.

8. (12 pts) Develop an algorithm that computes the $k^{th}$ largest element of a set of $n$ distinct integers in $O(n + k \log n)$ time.

9. (0 pts) What would the answer to this question be if you didn't already know the answer?