CS/COE 1501 Quiz 1

Name: **SOLUTION**

Pitt email: _____ @pitt.edu

Section (circle one): COE 1501; CS 1501 (writing); CS 1501 (non-writing)

Score: / 30

1. (3 points) f(x) is $\Omega(g(x))$ if:

   a) constants c_1, c_2, and x_0 exist such that $c_1 * |g(x)| <= |f(x)| <= c_2 * |g(x)| \ \forall x>x_0$

   b) constants c and x_0 exist such that $|f(x)| <= c * |g(x)| \ \forall x>x_0$

   c) constants c and x_0 exist such that $|f(x)| >= c * |g(x)| \ \forall x>x_0$

   d) None of the above

2. (4 points) Arrange the following orders of growth from least to greatest:

   2^n, n^4, log(n), n!, n

   log(n), n, n^4, 2^n, n!

3. (4 points) List and define 3 properties that we have used to compare sorting algorithms.

runtime: time required as input size increases

stability: does the algorithm maintain relative ordering of tied values?

is it in-place: does the algorithm require extra memory (space) to run?

4. (6 points) Show your work for sorting the following numbers in ascending order using radix sort.

247, 85, 87, 45, 210, 40, 95, 217, 80, 110

- 210, 40, 80, 110, 85, 45, 95, 247, 87, 217
- 210, 110, 217, 40, 45, 247, 80, 85, 87, 95
- 40, 45, 80, 85, 87, 95, 110, 210, 217, 247

5. (8 points) Describe how you would write a hash function to store street address keys (e.g., 123 Forbes Ave.) in a hash table.

Answers will vary, but the solution should use whole key, exploit differences between keys, and produce a uniform distribution of hash values.

Example:

• Treat the entire address as an ASCII string and use Horner's method to do modular hashing.

6. (5 points) Briefly describe how the Boyer Moore Mismatched Character Heuristic works. What is its runtime?

Preprocess the pattern to create an array (called the right array) that indicates the right-most position of each character in the pattern (all characters not in the pattern will get a value indicating that they're not in the pattern).

Then, to find the pattern within the text, start with the pattern at the beginning of the text, but compare characters left-to-right. When there's a mismatch, shift the pattern to the right however many positions are indicated in the right array. Use the mismatched character in the text when consulting the array; if it's not in the pattern, shift the pattern to the left so that it's completely past the mismatched character. Repeat until the pattern is found or the end of the text is reached.

Worst-case runtime: $\Theta(nm)$, where n is the size of the text and m is the size of the pattern.

Average-case runtime: $\Theta(n/m)$, where n is the size of the text and m is the size of the pattern.

(The student only needs to list one runtime, but they should label whether it's the worst-case or average-case.)