CMPSC 465
Data Structures and Algorithms

Spring 2013

Exam I: Introduction to Asymptotic Analysis and Divide-and-Conquer

January 31, 2013

Name: _____ Last 4 digits of ID: ____ ____ ____ ____

Honor Code Statement

I certify that I have not discussed the contents of the exam with any other student and I will not discuss the contents of this exam with any student in this class before it is returned to me with a grade. I have not cheated on this exam in any way, nor do I know of anyone else who has cheated.

Signature: _____

Directions/Notes:

- Write your ID on every page of this exam. Write your name just on this cover page.
- No outside notes, books, or calculators are permitted. However, you will be provided with a page giving a few useful bits of information.
- Be sure to sign the honor code statement when you are finished.
- All questions on this exam are implicitly prefaced with "As taught in CMPSC 465 lectures this term."
- Always justify your work and present solutions using the conventions taught in class.
- The problem that is a take-off on a challenge problem is on the last sheet of the exam. You must solve that problem first and remove and turn in that page of your exam within the first 30 minutes of the exam period. Due to the logistics of timing the challenge problem, if you arrive late without having made prior arrangements with the instructor, you will either forfeit missed time or be permitted to earn only up to half credit on the problem.
- Use pencil to complete this exam. Use of pen will result in an automatic 10-point deduction and we reserve the right not to read any problem with cross-outs.

Score Breakdown:

#	1	2	3	4	5	6	7	8	9	A.P.	Total
Score											
Value	15	10	10	7	6	10	12	10	20	--	**100**

1. **Ranking Growth of Functions.** Below are expressions that are functions of n. You have two jobs: First, as much as possible, simplify each expression and write it in terms of O-notation. Second, rank the functions in terms of asymptotic dominance. The fastest-growing function should be #1. If two functions grow at the same rate, they may share a ranking and the next ranking should be skipped. (For example, if you had a tie for 3rd place, you'd have rankings 1, 2, 3, 3, 5, 6, etc., but no 4.) [15 pts.]

function (and room for work)	**asymptotically simplified form**	**ranking**
$\frac{n\log_7 n}{5n}$		
$\sum_{i=0}^{n} 10^i$		
$\sum_{i=0}^{50} n^i$		
π^e		
$3(n^n - 3^n)$		
$\frac{35n^2 + 12n^3}{n(n+5)}$		
$3n + 150^n$		
$150 \cdot 3^n$		
$n^3 + \left(\frac{n}{7}\right)^{18} + log_{12} n + 18n!$		
$12n\log_2 n + n + 18n^2$		
$\log_7 n + 12n^5 - n^3$		
$12(\log_2 n + 1)$		
$\log_{10} n + 2013$		
$\frac{\log_3 n + n!}{17}$		

2. **Sorting**. [10 pts.]

a. Illustrate a complete trace of a merge sort of the array below, following the style used in lecture:

index	*1*	*2*	*3*	*4*	*5*
value	92	81	103	72	31

b. What is the theoretically predicted running time of merge sort? _____
[This question is just a recall question. No explanation necessary.]

c. Consider this claim: "For any array of size *n* where *n* is particularly large, it is guaranteed that merge sort will perform faster than insertion sort." Is this claim true or false? If it's true, give a *rigorous* argument why insertion sort will always perform worse than what you've claimed above. If it's false, describe the best scenario in which insertion sort would do better than what you've claimed above and rigorously derive the running time.

3. **Running Time of a Code Fragment.** Consider the following code fragment: [10 pts.]

```
b = 1
for i = 1 to n + 1
{
    a = 0
    for k = 2 to ⌊i / 2⌋
    {
        a = a + 1
    }
    b = b * (a + 5)
}
```

In terms of n, how many additions and multiplications (total) are performed when this code is executed with an **even** input of n? Simplify your result to a polynomial.

4. **Order Notation.** Using the formal definitions of order notation and theorems from class, prove the claims. [7 pts.]

   a. When $f(x) = 12x^6 - 6x^3 + 2x - 15$ and $g(x) = x^6$, $f(x)$ is $O(g(x))$.

   b. When $h(x) = 7x^3 + 2x^2 + 20$ and $g(x)$ is x^3, $h(x)$ is $\Omega(g(x))$.

   c. If we wanted to prove that $h(x)$ from (b) is $\Theta(g(x))$, what else would we need to do?
   [Don't actually prove this, but tell what you *would* do at a high level. This question is worth one point and it should take you less than 30 seconds to write down your answer.]

5. **Master Theorem.** Using the Master Theorem, give (and explain) tight asymptotic bounds for each recurrence. [6 pts.]

   a. $T(n) = 3T(n/9) + \sqrt{n}$

   b. $T(n) = 8\ T(n/2) + \Theta(n^2)$

6. **Closed Form of a Recurrence**. Consider the following recurrence: [10 pts.]

$a_1 = 12$

$a_k = 3a_{\lfloor k/2 \rfloor}$ for $k \geq 2$

Using some form of induction and the formal proof style used in class and Epp, prove that the closed form of this recurrence is $a_n = 12 \cdot 3^{\lfloor \lg n \rfloor}$. To save time, you only need to prove the inductive step for **odd** values of k.

7. **Designing an Algorithm.** Consider an array A containing n **distinct** integers. We define a local minimum of A to be an x such that $x = A[i]$, for some $0 \leq i < n$, with $A[i\text{-}1] > A[i]$ and $A[i] < A[i\text{+}1]$. In other words, a local minimum x is less than its neighbors in A (for boundary elements, there is only one neighbor). We want to solve the problem of finding **a** local minimum. Note that A might have *multiple* local minima, but you only need to locate and return *one*. [12 pts.]

   As an example, suppose A = [10, 6, 4, 3, 12, 19, 18]. Then A has two local minima: 3 and 18.

   a. Describe an algorithm using the divide and conquer mindset to solve this problem.

   b. Express a recurrence for the running time of your algorithm. Explain why this recurrence fits. [If you are unable to solve (a) and want some partial credit here, explain how to derive a recurrence from a divide-and-conquer algorithm in general and draw a large "X" over your work in (a).]

8. Use a recursion tree to guess a bound for the closed form of the recurrence $T(n) = 2T(n/5) + 3T(n/6) + cn$ that is as accurate as possible. You do not need to prove your closed form correct. [10 pts.]

9. **Algorithm Correctness [From Challenge Problem].** The following code fragment implements Horner's rule for evaluating a polynomial

$$P(x) = \sum_{k=0}^{n} a_k x^k$$
$$= a_0 + x(a_1 + x(a_2 + ... + x(a_{n-1} + xa_n)...))$$

given the coefficients a_0, a_1, …. a_n and a value for x:

```
y = 0
for i = n downto 0
    y = a_i + x · y
```

[20 pts.]

Note: This problem must be solved and turned in within the first 30 minutes of the exam period. Remove this sheet.

a. Consider the following loop invariant:

At the start of each iteration of the **for** loop,

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$$

Interpret a summation with no terms as equaling 0. Prove the correctness of this algorithm.

Initialization

Maintenance

Termination

b. What is the exact number of additions and multiplications performed by this code fragment? [Show work neatly.]

c. In Θ-notation, what is the running time?

Now consider the following pseudocode, which uses the same symbols as the problem statement and where methods have preconditions that $k \geq 0$ and $n \geq 0$:

```
EVAL-MONOMIAL(x, k)
    if k == 0
        return 1
    if k == 1
        return x
    else
        h = ⌊k / 2⌋
        b = EVAL-MONOMIAL(x, k mod 2)
        m = EVAL-MONOMIAL(x, h)
        return b · m · m
```

```
EVAL-POLYNOMIAL(a[0..n], x, n)
    sum = 0
    for k == 0 to n
        sum = sum + a[k] · EVAL-MONOMIAL(x, k)
    return sum
```

d. The book had you implement naïve polynomial evaluation, which ran in $\Theta(n^2)$ time. How does the performance of EVAL-POLYNOMIAL compare to naïve polynomial evaluation and Horner's rule? Explain. You do not need a rigorous argument, but should be able to see a bound on the running time of the above code and include that bound in your explanation.

IMPORTANT NOTES:

- This last problem must be solved in the first 30 minutes of the exam period.
- Remove this sheet. Listen for time to be called to pass it in.
- Make sure the last 4-digits of your ID number on this sheet match the rest of your exam or you will not get credit for this problem.
- Write the first letter (only) of your last name in the box to the right to assist us with sorting exams while relatively maintaining anonymity during grading.

First Letter of Last Name: