CSI 503 – Algorithms and Data Structures – Fall 2009
Midterm Examination I – October 1, 2009

Note: This exam has 4 questions for a total of 100 points. Answer all questions. Write all your answers on the blue books.

Question I (30 points total)
(a) Recall that a 3-input 1-output Boolean function takes a triple of Boolean values (True or False) as input and produces a Boolean value as output. Let \( C \) denote the set of all 3-input 1-output Boolean functions. Call a function in \( C \) special if it outputs False for the both the inputs (True, True, True) and (False, False, False). Assume that all the functions in \( C \) are equally likely. What is the probability that a randomly chosen function from \( C \) is special? Show work. (8 points)

(b) Let \( f(n) = \sqrt{n^{3/2} - n^{1/2}} + 1 \). Prove that \( f(n) = \Theta(n^{3/4}) \). Your answer must explicitly show the constants \( c_1, c_2 \) and \( n_0 \) such that \( c_1 n^{3/4} \leq f(n) \leq c_2 n^{3/4} \) for all \( n \geq n_0 \). You must also indicate how arrived at the values of those constants. (12 points)

(c) Suppose \( f(n) = n/\ln n \) and \( g(n) = n^{0.9} \ln n \). Prove by computing limits that \( g(n) = o(f(n)) \). (10 points)

Question II (25 points total)
(a) Suppose \( f(n) = 7n + 8 \) and \( g(n) = 2^{\log_2 n} \). Prove or disprove: \( g(n) = \Omega(f(n)) \). (6 points)

(b) Suppose \( n = 6^r \) for some non-negative integer \( r \) and \( T(n) \) satisfies the recurrence

\[
T(1) = 1 \quad \text{and} \\
T(n) = 5T(n/6) + n \quad \text{for all } n \geq 6.
\]

Use the iteration method to prove that \( T(n) = O(n) \). (10 points)

(c) Consider the following recursive function:

Function \( F(n) /* n \geq 1 \) is an integer; the function returns an integer. */

\[
\text{if } (n = 1) \text{ then return 1 else return } F(n - 1) + n * n * n
\]

For \( n \geq 1 \), let \( M(n) \) denote the number of multiplications used by the above algorithm for the input \( n \).

(i) Write the recurrence for \( M(n) \). (3 points)

(ii) Use the substitution method to show that \( M(n) = O(n) \). (6 points)
**Question III** (25 points total)

(a) Use the Master Theorem to find the asymptotic solution to the following recurrence:

\[ T(n) = 8T\left(\lfloor n/3 \rfloor\right) + 2n^2 \]

Show work. (13 points)

(b) Consider the following recurrence:

\[ T(n) = 4T\left(\lceil n/2 \rceil\right) + f(n) \]

Specify a function \( f(n) \) so that the asymptotic solution to the above recurrence is \( T(n) = \Theta(n^2\log n) \). For your choice of \( f(n) \), be sure to justify why the asymptotic solution is \( T(n) = \Theta(n^2\log n) \). (12 points)

**Question IV** (20 points total)

Let \( P[1..n] \) be an array each of whose elements contains an integer value (which may be positive, negative or zero). A subarray \( P[i .. j] \) of \( P \), where \( i \leq j \), consists of the elements \( P[i], P[i+1], \ldots, P[j] \). We say that subarray \( P[i .. j] \) is monotone if \( P[i] \leq P[i+1] \leq P[i+2] \ldots \leq P[j] \). Assume that a subarray consisting of just one element is monotone. This problem asks you to devise a divide-and-conquer based algorithm to determine the length of a longest monotone subarray of a given array.

**Example:** Consider the following array \( P[1 .. 8] \).

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
-1 & -2 & 9 & 6 & 7 & 8 & -19 & -18 \\
\end{array}
\]

In the above array, some of the monotone subarrays are \( P[2 .. 3], P[4 .. 6] \) and \( P[7 .. 8] \). For this example, the length of a longest monotone subarray is 3. (The corresponding subarray is \( P[4 .. 6] \).)

This problem has three parts.

(a) Indicate clearly what actions are to be performed in the Divide, Conquer and the Combine steps of your algorithm. The time taken by your Divide and Combine steps must be \( O(n) \). As part of the Conquer step, be sure to indicate when the recursion ends. (12 points)

(b) Based on your answer to Part (a), write the pseudocode for the function

\[
\text{int longest}(P, \ a, \ b)
\]

which returns the length of the longest monotone subarray of \( P[a..b] \). Assume that \( a \leq b \). (6 points)

(c) Let \( T(n) \) denote the running time of your algorithm when \( P \) has \( n \) elements. Specify a recurrence for \( T(n) \). Just give the recurrence – there is no need to solve it. (2 points)
Statement of Master Theorem:
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function and let $T(n)$ be defined on nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n)$$

where $n/b$ may be either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.

**Part 1:** If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.

**Part 2:** If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.

**Part 3:** If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$ and if $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and for sufficiently large $n$, then $T(n) = \Theta(f(n))$. 