CIS507: Design & Analysis of Algorithms
Quiz 1, Spring 2012

Duration: 15 minutes
Total weight: 5%

Student Name: –

Student ID: –

Problem	Points Obtained	Points Possible
1		1.5
2		1.5
3		2
Total		5

Cheat Sheet: Master Method

For $T(n) = aT(n/b) + f(n)$, with $a \geq 1$, $b \geq 1$, compare $f(n)$ with $n^{\log_b a}$.

Case	Condition, for $\epsilon > 0$	Solution
1	$f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$	$T(n) = \Theta(n^{\log_b a})$ Number of leafs dominates
2	$f(n) = \Theta(n^{\log_b a})$	$T(n) = \Theta(n^{\log_b a} \lg n)$ All rows have same asymptotic sum
3	$f(n) = \Omega(n^{\log_b a + \epsilon})$	$T(n) = \Theta(f(n))$ provided that $af(n/b) \leq cf(n)$ for some $c < 1$
2 (general)	$f(n) = \Theta(n^{\log_b a} \lg^k n)$ or some constant $k \geq 0$	$T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$ They grow at ‘similar’ rate

1 True or False (1.5 points)

1. **(0.5 point)** $2n^2 + 1,000n + 41 = \mathcal{O}(n^2)$

2. **(0.5 point)** Running merge sort on an array of size n which is already correctly sorted takes $\mathcal{O}(n)$ time.

3. **(0.5 point)** There exists functions $f(n)$ and $g(n)$ such that $f(n) = \mathcal{O}(g(n))$ and $f(n) = \omega(g(n))$.

2 Multiple Choice (1.5 points)

For each of the following, circle the correct answer(s). Note that there may be more than one in each question.

1. **(0.5 point)** Let $f(n) = \log(\sqrt{n})$ and $g(n) = \mathcal{O}(\log n)$.
   $f = \mathcal{O}(g)$ $f = \Theta(g)$ $f = \Omega(g)$

2. **(0.5 point)** Let $f(n) = \sqrt{n}$ and $g(n) = \log n$.
   $f = \mathcal{O}(g)$ $f = \Theta(g)$ $f = \Omega(g)$

3. **(0.5 point)** Let $f(n) = 5n \log n$, and $g(n) = n \log 5n$.
   $f = \mathcal{O}(g)$ $f = \Theta(g)$ $f = \Omega(g)$

3 Recurrences (2 points)

1. Give asymptotic upper and lower bounds for $T(n)$ for each of the following recurrences. Assume that $T(n)$ is a non-negative constant for $n \leq 10$. Make your bounds as tight as possible, and justify your answers.

   (a) **(0.5 point)** $T(n) = 3T(n/3) + n$ (use the Master Method)

   (b) **(0.5 point)** $T(n) = 11T(n/7) + n^3$ (use the Master Method)

2. **(1 point)** A brilliant scientist came up with a new variant of Mergesort. It splits an array with n elements into three parts, two of size $n/4$ and one of size $n/2$. After sorting the sub-arrays, it merges all three with just n comparisons. Write the recurrence for this algorithm.