Due Date: January 31, 2013

Problem 1: In each of the following cases, rank the functions by order of their growth. Here $\log n$ means $\log_2 n$.

- $2^{\log n}, (\log n)^{\log n}, e^n, 4^{\sqrt{\log n}}, n!, \sqrt{\log n}$
- $(\frac{3}{2})^n, n^3, (\log n)^2, \log(n!), 2^{2^n}, n^{\frac{1}{\log n}}, n^{\frac{1}{\log \log n}}$

Problem 2: Solve the following recurrences using induction and give a Θ bound for each of them and explain why.

- $T(n) = 2T(n/2 + 5) + n$
- $T(n) = 2T(n/2) + \frac{n}{\log n}$
- $T(n) = \sqrt{n}T(\sqrt{n}) + n$

Problem 3: Give an efficient algorithm to compute the *least common multiple* of two n-bit numbers x and y, that is, the smallest number divisible by both x and y. What is the running time of your algorithm as a function of n?

Problem 4: [Monge Arrays] An $m \times n$ array A of real numbers is a ***Monge array*** if for all i, j, k, and ℓ such that $1 \leq i < k \leq m$ and $1 \leq j < \ell \leq n$, we have

$$A[i,j] + A[k,\ell] \leq A[i,\ell] + A[k,j].$$

In other words, whenever we pick two rows and two columns of a Monge array and consider the four elements at the intersections of the rows and the columns, the sum of the upper-left and lower-right elements is less than or equal to the sum of the lower-left and upper-right elements.

(i) Here is a description of a divide-and-conquer algorithm that computes the leftmost minimum element in each row of an $m \times n$ Monge array A:

> Construct a submatrix A' of A consisting of the even-numbered rows of A. Recursively determine the leftmost minimum for each row of A'. Then compute the leftmost minimum in the odd-numbered rows of A.

Explain how to compute the leftmost minimum in the odd-numbered rows of A (given that the leftmost minimum of the even-numbered rows is known) in $O(m + n)$ time.

(ii) Write the recurrence describing the running time of the algorithm described above. Show that its solution is $O(m + n \log m)$.

Problem 5: Suppose we are given an array $A[1..n]$ with the special property that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that an element $A[x]$ is a *local minimum* if it is less than or equal to both its neighbors, or more formally, if $A[x-1] \geq A[x]$ and $A[x] \leq A[x+1]$. For example, there are six local minima in the following array:

9	7	7	2	1	3	7	5	4	7	3	3	4	8	6	9

We can obviously find a local minimum in $O(n)$ time by scanning through the array. Describe and analyze an algorithm that finds a local minimum in $O(\log n)$ time. *[Hint: With the given boudary conditions, the array must have at least one local minimum. Why?]*