CS61B, Fall 2008 **HW #6** **P. N. Hilfinger**

Due: Wed., 29 October 2008

Homework Exercises. You'll find a skeleton for your answers in the `hw6` staff directory. Put non-program answers into a file `hw6.txt` (plain text, please, no PDF, RTF, or Word files).

1. Suppose that $f(n)$ is a positive, non-decreasing function. Show that $\lceil f(n) \rceil \in O(f(n))$.

2. Suppose that $p(x)$ is any polynomial in x with positive coefficients. Show that $\log p(x) \in O(\log x)$.

3. Show that $\log_b f(x) \in \Theta(\log f(x))$ for any constant $b > 1$.

4. What is the worst-case running time for the following program fragment?

```
int j;
j = 0;
for (int i = 0; i < N; i += 1) {
   for ( ; j < M; j += 1) {
      if (bump (i, j))
         break;
   }
}
```

Assume that `M` and `N` are integer constants, and `bump` is a constant-time ($O(1)$) method that returns a boolean result. We're looking for a Θ result that uses `M` and `N`.

5. [Goodrich & Tamassia] Suppose that A is an $n \times n$ array of 1's and 0's with the property that all the 1's in a row come before all the 0's in that row. The array is actually huge ($n > 500000$), and instead of actually being stored as a Java array, it is represented by a `BitMatrix` object with a method `.get(`i, j`)`, which returns A_{ij} (i is the row, j the column). Fill in the method `mostOnes(A)` in the template file `BigMat.java` so that it returns the index of the row of A that contains the most 1's. When several rows contain the largest number of 1's, return the smaller index. Your method must operate in $O(n)$ time (*not* $O(n^2)$ time). Your program will be given a time limit that requires it to operate in better than $O(n^2)$ time.

6. Using the linked representation of binary trees in `Tree.java`, fill in the following complementary functions (see `FlattenTree.java`).

```
/** The sequence of all labels of T, in inorder. */
static String[] flatten (Tree T) {
   // FILL IN
}

/** A binary tree of minimum depth whose labels, in inorder, are
 *  L[0], L[1], .... */
static Tree treeify (String[] L) {
   // FILL IN
}
```

7. A *threaded tree* has an extra link in each node such that all of the nodes in the tree are linked into a list using this link. For example, you might use this link to store a pointer to the next node in preorder after this one, so that a preorder traversal of the tree is just a traversal of this list (see the example below; the threads are dashed.) Implement a function `preorderThread` that computes these links in a general tree. Use the template in file `ThreadedTree.java`.