Problem Set 2—Due Friday, October 11

Instructions: Write up your your solutions as clearly and succinctly as you can. Don't forget to acknowledge anyone with whom you discussed problems.

Problem 2–1. Refer to QUICKSORT and PARTITION on page 154 of your text. Recall that in class I had QUICKSORT(A, p, r) recursively call QUICKSORT$(A, p, q-1)$ (note the "-1") and QUICKSORT$(A, q+1, r)$. Routine PARTITION(A, p, r) is then expected to rearange the the elements of A so that everything to the left of $A[q]$ is less than or equal to $A[q]$, while everything to the right of $A[q]$ is greater than or equal to $A[q]$. The element at $A[q]$ (the "pivot") is, say, the element originaally at $A[p]$. Routine PARTITION in your book doesn't abide by this conventions. Make a *minimal* modification to PARTITION so that it does.

Problem 2–2. Do problem 7-1 on page 152 of your text.

Problem 2–3. You have n coins that are all supposed to be gold of the same wight. But you know that one coin is actually fake and weighs less than the others. You want to find it. You have a balance scale: you can put any number of coins on each side of the balance and it will tell you if the two sides weigh the same, or which side is lighter if they do not. Develop an algorithm to find the light coin in a minimum number of comparisons. Compute this minimum number of comparisons (exactly). How many weighing will it take you for $n = 50$ coins?

Problem 2–4. Let $(a_1, \ldots a_n)$ and $(b_1, \ldots, b_n)$ be lists, alread sorted into incresing order. Give an $O(\lg n)$-time algorithm which finds the median of the $2n$ elements $\{a_1, \ldots, a_n, b_1, \ldots, b_n\}$.

Problem 2–5. You are to sort a stack of n pancakes $p_1, \ldots, p_n$ so that the smallest is on the top and the largest is on the bottom. The only permitted operation is to insert the spatula and flip. For example, if you insert the spatula beneath pancake 1 and flip you get:

Before flip	After flip
2	1
5	3
3	5
1	2
7	7
6	6
4	4

Give the most efficient algorithm you can (as few flips as possible) to sort the stack of pancakes. Exactly state how many flips your algorithm uses as a function of the number n of pancakes.

Hint. My solution uses 0, 1, 3, 4 and $2n-4$ *flips when* $n = 1, 2, 3, 4$ *and* $n \geq 5$*, respectively.*