CS524
HW#2

Due: Monday, February 14

1. (8 points) (Adapted from Berman and Paul's *Algorithms*) A waiter has a stack of n disk-shaped pancakes of different sizes, and he wants to order them from largest on the bottom to smallest on the top. The operation he can perform is to grab the top m pancakes, $1 \le m \le n$, and flip them all over. Call this an *m-flip*. You may consider the pancakes to be labeled from 1 (the largest) to n (the smallest), and you may assume that n is even. So the final configuration is

$$\begin{array}{c} 1 \\ 2 \\ . \\ . \\ . \\ n \end{array}$$

a Prove an upper bound of $2n$-2 flips on the worst case complexity of solving this problem.
b Prove an lower bound of n-1 flips on the worst case complexity of solving this problem.

2. (3 points) Prove each of the following:
a $42 \in \Theta(1)$
b $\lg n \in \Theta(\log_{10} n)$
c $17n^3 \notin \Theta(400n^2)$

3. (14 points) ***a*** Describe an algorithm to find both the smallest and second smallest elements of a list of $A[1..n]$ distinct entries that does n-1 comparisons in the best case. Do not count the underlying comparisons implicit in the implementation of a control loop. For each n, what is a best case instance for your algorithm?
b How many comparisons does your algorithm use in the average case? Give a precise answer as a function of n; do not use Θ-notation. State clearly all assumptions.
c How many comparisons does your algorithm use in the worst case? What is a worst case instance for your algorithm?
d If A contains n distinct numbers and each input permutation is equally likely, then what is the probability that an input is a worst case instance?

4. (4 points) An *inversion* of an array is defined in **PROBLEM *2-4*** on pg. 39 of our text. Given an algorithm to sort array $A[1..n]$ in worst case time in $O(n+k)$, where k is the number of inversions of A.