Divide and Conquer / Sorting Advanced - Written Problems

1. You are interested in finding the median salary in *Polarized County*. The county has two towns, *Happyville* and *Sadtown*. Each town mantains a database of all of the salaries for that particular town, but there is no central database.

   Each town has given you the ability to access their particular data by executing *queries*. For each query, you provide a particular database with a value k such that $1 \leq k \leq n$, and the database returns to you the k^{th} smallest salary in that town.
   You may assume the following:

   - Each town has exactly n residents (so $2n$ total residents across both towns)
   - Every salary is unique (i.e., no two residents, regardless of town, have the same salary)
   - We define the *median* as the n^{th} highest salary across both towns

   Design an algorithm that finds the median salary across both towns in $\Theta(log(n))$ total queries.

2. State the complete recurrence for your algorithm. You may put your $f(n)$ in big-theta notation. Use the master theorem to show the solution for your recurrence is $\Theta(log(n))$. If your algorithm is slower, give us the actual running time.

3. Prove that your algorithm above finds the correct answer. *Hint: Do induction on the size of the input.*

4. Suppose you've been given the power to reschedule every major holiday of the year (and spread them out as you please). To aid with this process, you want to construct an algorithm that calculates how *balanced* a given schedule is. Let a schedule S be a list of n holidays given in chronological order they will be scheduled (For now, we only care about the relative order of the holidays, not the exact dates). Additionally, the list contains integers such that each $S[i]$ represents how many people love that holiday.

   You decide to define the *balance* of a schedule S as follows: Let H_L be the number of times a holiday is more popular than a holiday before it (or to the left of it in the list S). Likewise, let H_R be the number of times a holiday is more popular than a holiday after it (or to the right of it in S). Notice that a schedule in which all the most popular holidays are at the end of the year (like our calendar) has a large H_{left} value and a small H_{right} value. Balanced schedules have H_{left} and H_{right} values that are close to one another. Below is an example schedule and solutions:

$$S = [10, 2, 8, 4, 12]$$
$$H_L = 6$$
$$H_R = 4$$

H_L is 6 because 12 is larger than everything to its left, 4 is larger than 2, and 8 is larger than 2 (totaling 6). Likewise, H_R is 4 because 10 is higher than three things to its right plus 8 is larger than 4.

Construct an algorithm that is given a schedule S as input, and returns the H_L and H_R scores for that particular holiday schedule. Your algorithm must run in $o(n^2)$ time (that is little-oh, so your algorithm must be strictly more efficient than n^2). State your algorithm's running time.

5. Suppose that *Netflix*, for some reason, is worried about account sharing and comes to you with n total login instances. Suppose also that *Netflix* provides you with the means only to compare the login info of two of the items in the list. By this, we mean that you can select any two logins from the list and pass them into an *equivalence tester* which tells you, in constant time, if the logins were produced from the same account. However, you DO NOT have access to the account data itself (account numbers, etc.)

   You are asked to find out if there exists a set of at least $\frac{n}{2}$ logins that were from the same account. Design an algorithm that solves this problem in $\Theta(nlog(n))$ total invocations of the *equivalence tester*.