Introduction to Algorithms	Problem Set 5 (3 problems)
CS 4820 Fall 2018	Due 11:59pm Tuesday, October 2, no late days

Your homework submissions need to be typeset (hand-drawn figures are OK). See the course web page for suggestions on typing formulas. **There will not be late days allowed for this homework due to the upcoming prelim.**

The solution to each question needs to be uploaded to CMS as a separate pdf file. To help provide anonymity in your grading, do not write your name on the homework (CMS will know it's your submission). (Questions with multiple parts need to be uploaded as a single file.) Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

(1) Randomized Median Finding. In class, we saw the randomized median finding algorithm (section 13.5), and showed that its expected running time is $O(n)$ with n elements. In this problem, we will only focus on the comparisons the algorithm has to make.

Randomized median finding starts by picking a random element x and identifying its position i in the sorted list using with $n-1$ comparisons. If $n/4 \leq i \leq 3n/4$, we discard the part of the array that we know doesn't contain the median, which is at least $n/4$ elements, and recurse. If i is not in this middle range, we try again with a new random element. As shown in class and in the textbook, in expectation it takes two tries to find an element in the desired range (using $2(n-1)$ comparisons). Solving the recurrence $T(n) \leq 2(n-1) + T(3n/4)$, we find that the expected number of comparisons is at most $2n\frac{1}{1-3/4} = 8n$.

In this problem we will consider a small variation on this algorithm. Assume you proceed as follows: instead of picking 1 element at random, pick 3 different element of the array. Compare the three (all three comparisons), and pick the median x **of the three** selected elements. Now determine the position i of x in the array by comparing to the remaining elements not selected in the initial sampling step. As before if $n/4 \leq i \leq 3n/4$, we discard the part of the array that we know doesn't contain the median, which is at least $n/4$ elements, and recurse. If i is not in this middle range, we try again with picking new random elements.

(a) What is the probability that the index of the median of the three points, i, is in the middle range $n/4 \leq i \leq 3n/4$?

(b) **We suggest you only consider this part after the lecture on Monday, Sept 24th.** State the recurrence for this algorithm, and show that the expected number of comparisons made by this method is upper-bounded by cn for a $c < 6$.

(2) Finding plurality in a list. Suppose you are supervising voting for County Person of the Year at the county fair. At the fair, n voters each write on a handwritten slip of paper the name of their favorite candidate and put it in the ballot box. You will have to process these ballots manually; furthermore, because any resident of the county is a valid candidate, the total number of possible candidates m is actually almost as large as the total number of voters. Counting the votes received by every candidate will be resource-intensive and hard to do accurately.

However, you suspect that tracking all possible candidates won't be necessary. You think it is likely that either (a) one of the candidates were selected by a large plurality of the voters, in which you can declare a winner, or (b) two candidates obtained significant shares of the vote, in which case you can immediately perform a runoff election. The list of votes is written out on a "read-only" array for allowing audits later.

Setting the definition of "large plurality of voters" to 1/3 of the votes, find a deterministic algorithm running in $O(n \log n)$ time and using at most $O(logn)$ additional space (in addition to the read-only array) that returns all candidates that were selected by more than $n/3$ of the voters (there can be at most two of these). You may assume that n is a power of 2. (As always, include a proof of correctness and runtime complexity analysis.)

(3) Hashing. We suggest you only consider this problem after the lecture on Monday, Sept 24th. Hashing functions are defined as maps from a universe U to some target set $T = [0, \ldots, n-1]$. We define a *set* of hashing functions $\mathcal{H}$ as *universal* if for any two elements $u, v \in U$, if $u \neq v$, then for a random $h \in \mathcal{H}$, the probability that $h(u) = h(v)$ is at most $1/|T|$ (where T is the target set).

For each of the following we define some set of hashing functions $\mathcal{H}$ based on a protocol to choose a random function h. For each option, either prove that it is universal, or give a counter example that shows it is not (that is, give u and v with higher collision probability than $1/n$ when uniformly sampling $h \in \mathcal{H}$).

(a) Suppose we want the target set to be $[0, 1, \ldots, p-1]$ for a prime p, and suppose the universe is $U = [0, 1, \ldots, M]$ for $M >> p$. We propose the following random selection of the function: select an $a \in [0, \ldots, p-1]$ uniformly random, and let the function be $h_a(x) = (ax \mod p)$.

(b) Suppose we want the target set to be $[0, 1, \ldots, n-1]$, where n is not a prime, and suppose the universe is k-dimensional vectors $(x_1, x_2, \ldots, x_k)$, where each x_i is an integer in $[0, n-1]$. We propose the following random selection of the function: let p be a prime that is $p \geq n$, and select an $a_i \in [0, \ldots, p-1]$ uniformly random, let $a = (a_1, a_2, \ldots, a_k)$ and let the function be $h_a(x) = ((\sum_i a_i x_i \mod p) \mod n)$

(c) Suppose we want the target set to be $[0, 1, \ldots, p-1]$, where p is a prime, and suppose the universe is vectors $(x_1, x_2, \ldots, x_k)$ where each x_i is an integer in $[0, n-1]$. We propose the following random selection of the function: let p be a prime that is $p \geq n$, and select an $a_i \in [0, \ldots, p-1]$ uniformly random, let $a = (a_0, a_1, a_2, \ldots, a_k)$ and let the function be $h_a(x) = (a_0 + \sum_i a_i x_i \mod p)$