CS 161 (Stanford, Winter 2022) Homework 3

Style guide and expectations: Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we are looking for in homework solutions. We will grade according to these standards. You should cite all sources you used outside of the course material.
What we expect: Make sure to look at the "**We are expecting**" blocks below each problem to see what we will be grading for in each problem!

Exercises. The following questions are exercises. We suggest you do these on your own. As with any homework question, though, you may ask the course staff for help.

1 Exercise: Medians (4 pt.)

Given two arrays of length n, find the median of all elements of the two arrays.

1. If the arrays are unsorted, describe an algorithm that returns the median of the combined array and explain why it has the best possible runtime. **[We are expecting:** A runtime and a brief description of your algorithm.**]**
2. If the arrays are sorted, can you achieve a better runtime? **[We are expecting:** A short English description, Pseudocode, runtime analysis.**]**

2 Randomized Algorithms (12 pt.)

In this exercise, we'll explore different types of randomized algorithms. We say that a randomized algorithm is a **Las Vegas algorithm** if it is always correct (that is, it returns the right answer with probability 1), but the running time is a random variable. We say that a randomized algorithm is a **Monte Carlo algorithm** if there is some probability that it is incorrect. For example, QuickSort (with a random pivot) is a Las Vegas algorithm, since it always returns a sorted array, but it might be slow if we get very unlucky. We will revisit the Majority Element problem to get more insight on randomized algorithms.

Algorithm	Monte Carlo or Las Vegas?	Expected running time	Worst-case running time	Probability of returning a majority element
Algorithm 1				
Algorithm 2				
Algorithm 3				

[We are expecting: Your filled in-table, and a short justification for each entry of the table. You may use asymptotic notation for the running times; for the probability of returning a

majority element, give the tightest bound that you can given the information provided. Fill in the table below, and justify your answers.**]**

Algorithm 1: findMajorityElement1

Input: A population P of n elements
while *true* **do**
    Choose a random $p \in P$;
    if isMajority*(P, p)* **then**
        return p;

Algorithm 2: findMajorityElement2

Input: A population P of n elements
for *100 iterations* **do**
    Choose a random $p \in P$;
    if isMajority*(P, p)* **then**
        return p;
return $P[0]$*;*

Algorithm 3: findMajorityElement3

Input: A population P of n elements
Put the elements in P in a random order.;
`/* Assume it takes time` $\Theta(n)$ `to put the` n `elements in a random order */`
for $p \in P$ **do**
    if isMajority*(P, p)* **then**
        return p;

Algorithm 4: isMajority

Input: A population P of n elements and a element $p \in P$
Output: True if p is a member of a majority species
count $\leftarrow 0$;
for $q \in P$ **do**
    if $p = q$ **then**
        count ++;
if $count > n/2$ **then**
    return *True*;
else
    return *False*;

Problems. The following questions are problems. You may talk with your fellow CS 161-ers about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

3 Snakes on a Plane

A group of snakes are planning a trip and are now booking plane tickets. Each snake has a preferred airplane aisle in mind: left, middle, or right. You'd like to sort the snakes so that all the snakes that prefer the left aisle are on the left, all the snakes that prefer the middle aisle are in the middle, and all the snakes that prefer the right aisle are on the right. You can only do two types of operations on the snakes:

Operation	Result
`poll(j)`	Ask the snake in position j about its preferred airplane aisle
`swap(i,j)`	Swap the snake in position j with the snake in position i

However, to do either operation, you need to pay the snakes to co-operate: each operation costs one stale hot dog. Also, you didn't bring a piece of paper or a pencil, so you can't write anything down and have to rely on your memory. Like many humans, you can remember up to seven integers between 0 and $n-1$ at a time.

3.1 (10 pt.)

Design an algorithm to sort the snakes which costs $O(n)$ hot dogs, and requires you to remember no more than seven integers between 0 and $n-1$ at a time. **[We are expecting:** Pseudocode **AND** a short English description of your algorithm.**]**

3.2 (5 pt.)

Justify why your algorithm is correct, why it takes only $O(n)$ hot dogs, and why it requires you to remember no more than seven integers at a time. **[We are expecting:** Informal justifications of the correctness, runtime, and memory usage of your algorithm that are both clear and convincing to the grader. If it's easier for you to be clear, you can give a formal proof of correctness, but you do not have to. It is okay to appeal to the correctness of an algorithm that we have seen in class, as long as you clearly explain the relationship between the two algorithms. **]**