CS 161
Summer 2018

Homework 0
Due: **This is worth 0% of your grade**

Exercises

Exercises should be completed **on your own**.

---

0. Install Python 3. Then go to `jupyter.org` and follow the instructions to install Jupyter notebook. If you are having difficulty, please talk to a TA during office hours or ask on Piazza.

1. See the Jupyter notebook `hw0.ipynb` for Exercise 1. This file includes the function `estimateMean`, which we have reproduced here.

```python
def estimateMean(A):
    samples = []
    # Draws a random sample of 10 elements with replacement from A
    for i in range(10):
        samples.append(A[random.choice(range(len(A)))])
    # Returns the sample mean
    return sum(samples) / len(samples)
```

   (a) `estimateMean(A)` attempts to estimate the mean of an array of numbers A. Show that the expected value that `estimateMean(A)` returns is indeed the mean of A.

   [We are expecting: A formal proof.]

   (b) In the notebook, there is some code for trying out `estimateMean(A)` a bunch of times for lists with elements between 0 and 30, and which plots the error. Based on playing around with this code, is it likely that the estimate returned by `estimateMean(A)` is off by more than 20? How likely or unlikely is this? Does your answer depend on n?

   [We are expecting: Your answers to the questions along with a convincing argument (a plot is fine; a formal proof is not required).]

Problems

You can collaborate with your classmates about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your solutions 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.

---

1. **(Minimum finding)** Given a zero-indexed array A_1 of n integers, we say a second array A_2 is a *rotation* of the first if there exists a *pivot* $p \in \{0, \ldots, n-1\}$ such that A_2 can be rewritten as the concatenation of two subarrays of A_1 as follows: $[A_1[p], \ldots, A_1[n-1]] + [A_1[0], \ldots, A_1[p-1]]$, where $+$ is the concatenation operator. For example, if $A_1 = [1,2,3,4]$, then there are four valid rotations: $[1,2,3,4]$; $[2,3,4,1]$; $[3,4,1,2]$; and $[4,1,2,3]$.

   (a) Design a simple $O(n)$-time algorithm to find the minimum element of an array A, where A is a rotation of a sorted array (in ascending order). You can assume A does not contain any duplicate elements. For example, if $A = [11, 13, 17, 23, 2, 3, 5, 7]$, then your algorithm should return 2.

   [We are expecting: Pseudocode and a brief English description.]

   (b) Design an $O(\log(n))$-time divide-and-conquer algorithm to find the minimum element of an array A, where A is a rotation of a sorted array (in ascending order). Again, you can assume A does not contain any duplicate elements.

   [We are expecting: Pseudocode and brief English description, as well as an informal justification of the running time. You do not need to prove that your algorithm is correct.]

   (c) In the previous two parts, you could assume A did not contain any duplicate elements. What is an example of an array A containing duplicate elements that is still a rotation of a sorted array, but causes your algorithm from part (b) to return a non-minimum element?

   [We are expecting: An example array.]