Lab 4

1. A 10-sided die—having values 1, 2, . . ., 10 on its faces—is tossed repeatedly.
    a. What is the expected number of tosses required to get a 2?
    b. What is the expected number of tosses required to get a total of three 2's?

    Assume that each of the values 1, 2, . . ., 10 is equally likely to appear. Explain your answer.

2. Is there a comparison-based sorting algorithm which, when run on an array containing exactly 4 distinct integers, requires only 3 comparisons to sort the integers, even in the worst case? Is there a comparison-based sorting algorithm which, when run on an array containing exactly 4 distinct integers, requires only 4 comparisons to sort the integers in the worst case? Prove your answers.

3. Goofy has thought of a new way to sort an array `arr` of n distinct integers:
    a. Step 1: Check if `arr` is sorted. If so, return.
    b. Step 2: Randomly arrange the elements of `arr` (using your work in Problem 2 of this lab)
    c. Step 3: Repeat Steps 1 and 2 until there is a return.

    Answer the following:

    A. Will Goofy's sorting procedure work at all? Explain
    B. What is a best case for GoofySort?
    C. What is the running time in the best case?
    D. What is the worst-case running time?
    E. What is the average case running time?
    F. Is the algorithm *inversion-bound?*

    Hint for Part E: What is the probability that a randomly arranged array of integers happens to be in sorted order? Then think about the coin-flipping model introduced in the slides.

4. In our average case analysis of QuickSort, we defined a *good self-call* to be one in which the pivot *x* is chosen so that number of elements < x is less than 3n/4, and also the number of elements > x is less than 3n/4. We call an x with these properties a *good pivot.* When n is a power of 2, it is not hard to see that at least half of the elements in an n-element array could be used as a good pivot (exactly half if there are no duplicates). For this exercise, you will verify this property for the array A = [5, 1, 4, 3, 6, 2, 7, 1, 3] (here, n = 9). Note: For this analysis, use the version of QuickSort in which partitioning produces 3 subsequences *L, E, R* of the input sequence *S.*
    a. Which x in A are good pivots? In other words, which values x in A satisfy:
        i. the number of elements < x is less than 3n/4, and also
        ii. the number of elements > x is less than 3n/4
    b. Is it true that at least half the elements of A are good pivots?
5. Devise an algorithm that performs *sideways sorting* on the elements of a length-n integer array. When an array is sideways-sorted, the elements are arranged as follows:
    position 0: the smallest integer

position 1: the largest integer
position 2: the second smallest integer
position 3: the second largest integer etc.

For example, when you sideways sort the input array {1, 2, 17, -4, -6, 8} you get: {-6, 17, -4, 8, 1, 2}. (Notice that –6 is the smallest, 17 the largest, -4 second smallest, 8 second largest, etc.) Answer the following:

A. What is the asymptotic running time of your algorithm?
B. Prove that it is *impossible* to obtain an algorithm to do sideways sorting of an integer array that runs asymptotically faster than the algorithm you created in Part A.