Lab 3

1. Suppose an array of length n is populated randomly with the letters A and B, with the guarantee that both letters occur at least once and that A and B are equally likely to occur. Let `Search(S, x)` be the usual algorithm for searching for an element `x` in an array `S`, which returns the position of the first occurrence of `x` in `S` (or –1 if not found). Prove that the average-case running time for `Search(S,x)` is < 2 whenever x is either A or B, and therefore `Search` (for A or for B) has average case asymptotic running time O(1).

2. A company uses a well-known sorting algorithm to sort its data. A best case for this sorting algorithm occurs when its input is an already-sorted array. In such cases, it runs in O(n) time. A worst case occurs when its input is reverse-sorted. In that case, it runs in O(n^2) time. The company knows from experience that all input arrays are either sorted or reverse sorted, but nearly all input arrays are already sorted. In fact, it is estimated that, for any collection of n arrays from the pool of all length-n arrays in the company's data store, only one of these arrays is ever reverse-sorted. What is the average-case asymptotic running time of the algorithm, given this distribution of inputs? Prove your answer. *Hint*: Review the Lesson 3 slides.

3. A die is tossed repeatedly.
   a. What is the expected number of tosses required to get a 6?
   b. What is the expected number of tosses required to get a total of three 6's?

   In each case, prove your answer.

4. Design an algorithm that does the following: Input is a set S of n integers together with an integer k. Your algorithm outputs "true" if there is some subset of S, the sum of whose elements is exactly k; it outputs "false" if no such subset can be found. What is the asymptotic running time of your algorithm? Explain.

5. Goofy has thought of a new way to sort an array `arr` of n distinct integers: Create a `temp` array and copy arr into temp.
   a. Step 1: Check if `temp` is sorted. If so, return.
   b. Step 2: Randomly permute the elements of arr and place the result in `temp`.
   c. Step 3: Repeat Steps 1 and 2 until there is a return.

   Will Goofy's sorting procedure work at all? What is a best case for GoofySort? What is the running time in the best case? What is the worst-case running time? What is the average case running time?

6. Recall the recursive algorithm for computing the nth Fibonacci number and the fact that it runs in exponential time. Improve this algorithm by thinking of the algorithm as a procedure to solve many subproblems, and to put the solutions to these subproblems together to obtain a final solution. For computing the nth Fibonacci number, the subproblems are computations of the mth Fibonacci number for each m < n. Putting these together allows the algorithm to compute the nth Fibonacci number. The problem with the recursive algorithm is that it *re-computes* solutions to these subproblems over and over again.

In this problem, re-work the recursive Fibonacci algorithm fib(n) so that, for each $m < n$, when fib(m) is computed for the first time, the return value is stored (in an array, for example), and when this value is needed at a later stage, instead of re-computing, the algorithm simply reads the stored value. (Storing values of solutions to subproblems is called *memoization.*) What is the running time of your new algorithm?