APS Homework 1: Divide-and-Conquer
Optional Challenge Problems

Problem 1: Efficient Power Function

Let $power(a,b)$ denote the power function a^b. For example, $power(5,3) = 5^3 = 125$. A trivial algorithm to implement $power(a,b)$ is the following:

```
Algorithm power(a,b):
    result ← 1
    Repeat b times:
        result ← result × a
    Return result
```

Note, however, that we will always have to perform exactly b multiplication operations. Can we compute $power(a,b)$ using fewer than b multiplication operations?

Problem 1a: Describe a Divide-and-Conquer algorithm for computing $power(a,b)$ using the fewest number of multiplication operations possible. You may assume that b is an integer.

Problem 1b: Prove that the algorithm you provided in *Problem 1a* is correct for any number a and for any integer b.

Problem 1c: As a function of a and/or b, what is the minimum number of multiplication operations needed to compute $power(a,b)$?

Problem 2: Counting the Number of 0s in a Sorted Binary List

A "binary list" is a list containing only 0s and 1s. For example, [0, 1, 0, 0, 1] is a binary list. A "sorted binary list" (in ascending order for this problem) is a binary list such that no 1s are ever followed by 0s. For example, [0, 0, 0, 1, 1] is a sorted binary list. Given a sorted binary list containing n elements, we can determine the number of 0s as follows:

```
Algorithm count_0s(nums):
    For i from 0 to |nums|-1:  # using 0-based indexing
        If nums[i] is 1:
            Return i + 1
    Return |nums|
```

However, for a list containing n elements, this simple algorithm requires us to look at all n elements in the worst-case scenario (all elements are 0s), which can become slow as n becomes large. Can we compute the number of 0s in a sorted binary list by looking at fewer than n elements?

Problem 2a: Describe a Divide-and-Conquer algorithm for determining the number of 0s in a sorted binary list by looking at the fewest number of elements possible.

Problem 2b: Prove that the algorithm you provided in *Problem 2a* is correct for any arbitrary sorted binary list containing $n > 0$ elements.

Problem 2c: As a function of n, what is the minimum number of elements needed to be looked at to determine the number of 0s in a sorted binary list of length n?

Problem 3: Tower of Hanoi

"Tower of Hanoi" is a game in which you have 3 pegs (call them *left*, *middle*, and *right*) and a tower of different-sized rings on *left* (Fig. 1). The rings are stacked such that, from bottom-to-top, they go from largest-to-smallest.

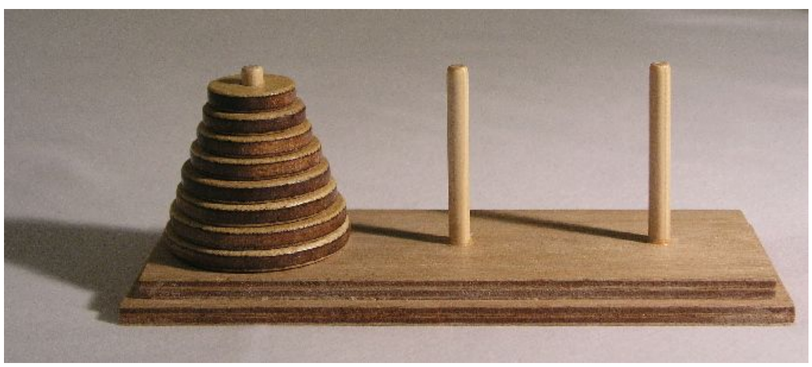

Figure 1. The "Tower of Hanoi" game.

The goal of the game is to move the entire tower from *left* to right, but with the following constraints: (1) you must move a single ring from one peg to another in any given move, and (2) you cannot place a ring on top of another ring smaller than itself (but you *can* place a ring on top of another ring *larger* than itself). In addition to simply moving the tower from *left* to *right*, you also want to try to minimize the number of moves.

Problem 3a: What is a solution for winning the "Tower of Hanoi" game given a tower containing $n = 3$ rings? What about $n = 4$ rings?

***Problem 3b*:** What is the minimum number of moves needed to win the "Tower of Hanoi" game given a tower containing $n = 3$ rings? What about $n = 4$ rings?

***Problem 3c*:** Describe a Divide-and-Conquer algorithm for winning the "Tower of Hanoi" game given a tower containing $n > 0$ rings.

***Problem 3d*:** Prove that the algorithm you provided in *Problem* 3c is correct for any arbitrary $n > 0$.

***Problem 3e*:** As a function of n, what is the minimum number of moves needed to win the "Tower of Hanoi" game given a tower containing $n > 0$ rings?