Recursion Extension Problems

- Efficiency/time complexities for the topic are considered. Note that these discussions are not exhaustive or representative of the types of questions that may be found on the exams.
- Practice problems are given to test your understanding of the topic. These problems are not for a grade, but promote a deeper level of understanding of the topic. Unless otherwise specified, assume you are using *n* data/objects/elements in the data structure or algorithm.
- We look at the worst/average/best case of Big O; however, do not expect all three cases to be explored extensively.
- Feel free to discuss these problems in recitation or during office hours at the TA Help Desk.

Extension

Coding

Implement each of the below functions recursively. Do NOT use any loops for problems, but be sure to use recursion for all problems! Always keep in mind efficiency; always try to code a more efficient solution.

1) Write a method called fibonacci() that accepts a positive integer, n, and returns the *n*th fibonacci number in the sequence. The fibonacci sequence begins with 1 and 1, and each number thereafter is the sum of the previous two numbers in the sequence. Ffibonacci sequence: 1, 1, 2, 3, 5, 8, 13, etc… . Throw an IllegalArgumentException if the number passed into the method is not positive or an integer.

2) Write a method called printHashSymbols() that takes in a non-negative integer, *n*, and prints out 3^n number of pound symbols, #. You may not use Math.pow for this method. Throw an IllegalArgumentException if n is negative or not an integer.

3) Write a method called printAddSub() that takes in a positive integer, *n*, and prints out a total of "*n*" addition and subtraction symbols (+ and -). If "*n*" is even, then print only 2 subtraction symbols placed together. If "*n*" is odd, then print only 1 subtraction symbol. The remainder of the symbols to be printed are addition symbols. The addition symbols are to be evenly distributed on both sides of the subtraction symbols. Throw an IllegalArgumentException if n is not positive or an integer. Sample output of function calls:

    printAddSub(1): -
    printAddSub(2): --
    printAddSub(3): +-+
    printAddSub(10): ++++--++++
    printAddSub(11): +++++-+++++

4) Write a method called sumOfEvens() that takes in a non-negative integer, *n*, and returns the sum of the first "*n*" positive even integers. Throw an IllegalArgumentException if *n* is negative or not an integer. Avoid using shortcuts for this method, such as using $n^2 + n$. Sample output of function calls:

    sumOfEvens(5) returns 30, since 2 + 4 + 6 + 8 + 10 = 30