CS:5350 Homework 1, Spring 2016

Due in class on Thu, Jan 28

Collaboration: You are welcome to form groups of size 2 and work on your homeworks in groups. Of course, you are not required to work in groups. Every group should make one submission and names of both group members should appear on the submission and both students in a group will receive the same score. Other than the TA and the professor, you can only discuss homework problems with your group partner. Collaboration can be positive because talking to someone else about these problems can help to clarify your ideas and you will also (hopefully) get to hear about different ways of thinking about the problem. On the other hand, collaboration can be negative if one member of the group rides on work being done by the other member – please avoid this situation. If your solutions are (even partly) based on material other than what has been posted on the course website, you should clearly acknowledge your outside sources.
Late submissions: No late submissions are permitted. You will receive no points for your submission if your submission is not turned in at the beginning of class on the due date.
Evaluation: Your submissions will be evaluated on correctness *and* clarity. Correctness is of course crucial, but how clearly you communicate your ideas is also quite important.

1. Consider the following recursive function to compute the n^{th} *Fibonacci number*.

```
function fibonacci(n):
    if n = 1 or n = 2 then
        return 1
    else
        return fibonacci(n-1) + fibonacci(n-2)
```

   (a) Let $A(n)$ denote the number of addition operations performed by a call to `fibonacci(n)`. Since no additions are performed when $n = 1$ or $n = 2$, we see that $A(1) = A(2) = 0$. Write down a recurrence relation for $A(n)$ for $n > 2$.

   (b) Implement a function in your favorite high level programming language (`Java`, `Python`, `C`, `C++`, `Scheme`, `Scala`, etc.) that computes $A(n)$. In other words, write a function called `numAdditions` that takes a positive integer n as argument and returns $A(n)$. Use your implemented function to figure out what $A(200)$ is.

2. Problem 15(a) from Chapter 1 (Recursion) on Page 20 from Jeff Erickson's notes.
   Hint: Reduce this problem to a problem we have discussed in class. Your solution to this problem should simply be a clear and precise description of the reduction.

3. Problem 11(a), (b), and (c) from Chapter 1 (Recursion) on Page 18 from Jeff Erickson's notes.
   Notes on 11(a): For any element a in $A[0..n-1]$, let the $rank(a)$ denote the position of a in the correctly sorted version of $A[0..n-1]$. You want to make certain claims (and prove them!) about where elements are based on their ranks after each call to StoogeSort. For example, you want to show that all elements of rank more than $m-1$ are in positions $n-m$ through $n-1$ after the first call to StoogeSort.