# CS100J    Lab 06. Recursion    Fall 2006

Name ____________________    Section time ______________    Section instructor __________________

In this lab, you will gain experience with writing recursive functions. Remember that creating/understanding a recursive function involves four points.

1. A precise specification of the function. Without this, you cannot write the function.
2. Handling the base case properly. The base case involves the "smallest" parameter values, for which the result can be given easily, without the need for recursive calls. For a function that works on the natural numbers `0, 1, 2, ...`, the base case is `0`, or `0` and `1`, usually. For a function that works on a `String`, the base case is the `""` or a `String` of length 0 or 1, usually.
3. Handling the recursive case properly. Solve the original problem in terms of the same problem but on a smaller value. For example, if the function involves a `String s`, the solution should be describable in terms of the same problem on a substring of `s`. In writing/understanding a recursive call, understand it in terms of the *specification* of the function being called.
4. In keeping with point 3, in a recursive call, the arguments must be in some sense smaller than the parameters of the method; this is what ensures termination. Each recursive call has smaller arguments, so that after some point, the base case will be reached.

On your computer, create a new folder. In DrJava, start a new class named `Rec` (for Recursion) and make sure that it is saved in the new folder. Also, start a JUnit test class called `RecTester`. As you develop the functions below, create the necessary test cases to test them and place appropriate `assertEquals` procedure calls in `RecTester`. Test each function this way before going on to the next one.

Because you are new to recursion, we will help you out by giving you a skeleton with all the methods headers and method specifications filled in for you —in file Rec.java on the course website. But please remember that you couldn't write these methods without a complete specification! When *you* design your own methods, *write the specifications first*. You may not finish these during the lab. If not, finish them at home and show the file to your TA the next week.

1. Count the number of characters in a String s that are not a given character c.

2. Produce a copy of a string with adjacent duplicates removed.

3. Produce a copy of String s with the first occurrence of character c removed (if present).

4. Produce a copy of String s with all occurrences of character c removed.

5. Produce the reverse of a String. For example, for the String "abcd", the result is "dcba". We give hints.

6. Compute Fibonnaci number n. The Fibonnaci numbers are 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... Fibonnaci number 0 is 0, Fibonnaci number 1 is 1, Fibonnaci number 6 is 13. Each number (except the first two) is the sum of the two previous ones.

7. Computer b to the power c, for $c \geq 0$. We give hints.