CIS 122 Summer 2019
Challenge Problems – Set 2

Challenge problems will not be guaranteed to follow our material exactly – meaning that solutions to these problems may involve concepts or techniques that we haven't seen yet, or may not formally cover at all. Additionally, these problems are neither required nor testable material.

This week's theme: recursion.
Recursion, generally speaking, is the property of self-similarity. It arises in nature all the time, in the form of fractal structures. In computing, "recursion" is used to refer to a function that, in the course of its execution, makes another call *to itself.* The problems that can be solved with recursive functions are problems that can be defined in terms of *smaller instances of the same problem.* For example, how do you count from 10 to 0? You say "10", and then you count from 9 to 0 (← smaller instance of same problem). How do you compute the factorial of 10? You multiply 10 by the factorial of 9 (thus factorial of 9 is a smaller instance of same problem).

Finally, a recursive function must hit what is called a "base case" – a condition in which we STOP recursing, we stop opening new function calls. Without the base case, we would recursive infinitely. The base case is simply the *trivial* instance of the problem. For example, counting down from 0, or computing the factorial of 0 (which is defined to be 1). A recursive function usually checks whether it is an instance of the base case before proceeding to recursively call itself. If it has reached the base case, it simply returns the answer.

Use whatever resources necessary to get started. When you have some ground underneath you, try to write the recursive string reverse with no outside help.

In all of these cases, start by identifying 1) the base case of the problem, and 2) the recursive relationship inherent in the problem. In other words, how can this problem be written in a way that includes a smaller instance of itself?

1) Recursive countdown

Write a recursive function, countdown(from), which counts from the single parameter from, all the way to and including 0 (simply printing these numbers is sufficient). This means that countdown must call itself.

2) Recursive factorial

Write a recursive function factorial_r(num) that computes the factorial of num, recursively.

3) Recursive reverse

Write a recursive function str_reverse_r(string), which returns the reverse of string.