UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

CSE 15: Discrete Mathematics
Laboratory 5

Spring 2019

Introduction

This lab is all about using recursion as a problem solving tool. All solutions you provide *must* be recursive. Download the file `recursion.py` from the folder associated with this lab on CatCourses. It contains definitions, and test cases for several functions. None of the functions in the file have been implemented. Your task is simple, implement the functions and upload your completed `recursion.py` file to the appropriate CatCourses assignment. Once again, your solutions have to be recursive. This means no loops of any kind. Points will be deducted if your solutions are not recursive or if you use built-in libraries to implement your functions. So, once again, everything from first principles and recursive.

Exercises

At the start of the file `recursion.py`, there is a call to set the recursion depth limit to 100000. Python typically sets this to 1000, which is too small. The exercises for this lab are as follows:

1. `factorial(n)` - This is the factorial function, which we have seen in class.
2. `fib(n)` - This is the Fibonacci number function. Each Fibonacci number is defined as the sum of the previous two Fibonacci numbers. The zeroth Fibonacci number is 0, and the first one is 1.
3. `equal(A, B)` - This function checks if two strings are equal. They have to be exactly the same. Your solution must be recursive. So do not just say `return A == B`.
4. `addup(list)` - This function takes as input a list of numbers and adds them all up. Once again, the solution must be recursive.
5. `reverse(A)` - This function takes in a string and returns the string in reverse. As before, recursive solutions only, no loops.
6. `isSorted(list)` - This function takes in a list and checks to see if it is sorted in ascending order (smallest to largest). This is getting a bit repetitive but no loops, recursion only.
7. `linearSearch(A, value)` - This function takes in a list and a value and returns `True` if the value is in the list, and `False` otherwise.
8. `binarySearch(A, value)` - This algorithm is a divide-and-conquer algorithm. It takes in a sorted list and a value. It then looks for the value at the middle of the list. If the value it is looking for is smaller than the middle element, it looks for it again but in the first half of the list only, otherwise in the second half only.