Recursion Muscles

In this lab, you will write a few Python functions using recursion. You must only use recursion, conditional statements (`if, else, elif`), list or string indexing and slicing. Some of these problems can be written without using recursion, e.g. using `map`, `filter`, `reduce`, `filter` or other looping structures. However, the objective here is to build your recursion muscles, so only stick to recursion. There will be a deduction for using any loops.

Do not use built-in functions (e.g. `len`, `sum`, etc.). However, your functions may call other functions that you write yourself.

Please be sure to name your functions exactly as specified for grading purposes.

• **dotProduct(L, K)** should output the dot product of the lists `L` and `K`. Recall that the dot product of two lists is the sum of the products of the elements in the same position in the two list.

You may assume that the two lists are equal in length. If these two lists are both empty, the output should be 0.0. Assume that the input lists contain only numeric values.

```
>>> dotProduct([5,3], [6,4]) <-- Note that 5*6 + 3*4 = 42
42
```

• **removeAll(e, L)** takes in an element `e` and a list `L`. Then, `removeAll` should return another list that is identical to `L` except that all elements identical to `e` have been removed. Notice that `e` has to be a top-level element to be removed, as the examples illustrate:

```
>>> removeAll(42, [ 55, 77, 42, 11, 42, 88 ])

[ 55, 77, 11, 88 ]

>>> #NOTICE BELOW THAT 42 IS NOT TOP-LEVEL
>>> #IT'S "HIDING" DEEP INSIDE OTHER LISTS AND
>>> #IS NOT ITSELF AN ELEMENT OF THE LIST L
>>> removeAll(42, [ 55, [77, 42], [11, 42], 88 ])

[ 55, [77, 42], [11, 42], 88 ]

>>> # NOTICE BELOW THAT THE LIST [77, 42] IS TOP-LEVEL
>>> # IT'S AN ELEMENT OF THE LIST L
removeAll([77, 42], [ 55, [77, 42], [11, 42], 88 ])

[ 55, [11, 42], 88 ]
```

• **geometricSeq(n, f, b)** takes in elements n, f, and b, where n is the nth number in the sequence, f is the multiplicative factor, and b is base case for when n=1. This function solves the formula a(n) = f×a(n-1) for all n >= 1. Here are some examples:

```
>>> geometricSeq(1, 2, 5)
5

>>> geometricSeq(3, 3, 1)
9

>>> geometricSeq(3, 2, 10)
40
```

• **deepReverse(L)** takes as input a list of elements where some of those elements may be lists themselves. `deepReverse` returns the reversal of the list where, additionally, any element that is a list is also `deepReverse`d. Here are some examples:

```
>>> deepReverse([1, 2, 3])

[3, 2, 1]

>>> deepReverse([1, [2, 3], 4])
[4, [3, 2], 1]

>>> deepReverse([1, [2, [3, 4], [5, [6, 7], 8]]])
[[[8, [7, 6], 5], [4, 3], 2], 1]
```