CCPS506 Lab 4 – Elixir: Tail recursion

Preamble

In this lab you'll practice writing recursive functions in Elixir. Each of these questions would be very easy in Java using arrays and loops, but in functional languages recursion is the name of the game.

Lab Description

Create an Elixir source file called **`Lab4.ex`**. In this file you will define a module named **`Lab4`**. For the problems below, you will implement your own tail recursive functions. No built-in shortcuts are permitted, no indexing, no iteration. Your functions must call themselves in some recursive fashion. Unless otherwise specified, input lists may contain any combination of types, and be of arbitrary length. **Assume nothing!**

i) **`sumEven` –** Return the sum of all **even integers** in an input list.

ii) **`sumNum` –** Returns the sum of all **numeric values** in the list.

iii) **`tailFib` –** accepts an integer argument, **n**, and returns the nth Fibonacci number. Assume the first two Fibonacci numbers are 1 and 1. That is, tailFib(1) == 1, tailFib(2) == 1. There is no tailFib(0). Your tail-recursive implementation must avoid the double recursive call. No $O(2^n)$!

iv) **`reduce` –** Consider MyEnum.reduce that we saw in class. Lab4.reduce will build on this. Your implementation for Lab4.reduce must include the two-argument version we saw in class, but you will also add an implementation that handles the optional 3rd argument to initialize acc. For example – the following two calls, exactly as written, should work correctly:

```
Lab4.reduce([1, 2, 3], fn(x, acc) -> x+acc end)
Lab4.reduce([1, 2, 3], 10, fn(x, acc) -> x+acc end)
```

You may use helper functions if you wish, so long as the user can invoke your function as seen above. Also remember that the input list need not be numeric! Lab4.reduce should work on a list of strings, for example, so long as the first-class function argument would work on strings. ***Hint:*** *Don't naively initialize the accumulator to zero…*

Submission

Labs are to be completed and submitted *individually*. Submit your **Lab4.ex** file containing the Lab4 module and all completed functions on D2L, under the submission for Lab 4.