HW 7

Due Mar 3 by 11:59pm **Points** 25 **Submitting** a file upload

1. (5 points) Take a look at problems 3 and 5 from HW 6. Do these again, but this time you MUST return an anonymous function (no named inner function). If you used an anonymous function for HW 6, just resubmit your work. You may NOT use a helper function with extra parameters.
2. Classes (15 points). Implement the class Biguint (you may remember this class from CSCI 60). Sometimes you need to store reaaaaaally big integers that would cause an Int to overflow. So we're going to create a class Biguint (for Big Unsigned Integer) to allow us to store arbitrarily large unsigned integers. The class will have a member integer list, where each spot in the list stores one digit of the number. You will store the 1s digit in the first position, the 10s digit in the second position, the 100s digit in the third position, etc. That is, the number 1,472 would be stored in the list: 2::7::4::1::Nil.

- For your class, the primary constructor will take in a list of integers, and simply initialize the member variable. You may make a public val for the list if you like as well.
- Also include the following constructor that takes in a string; for example 1,472 would be represented as "1472" (this allows us to represent integers with an arbitrary number of digits).
  def this(s:String) = this({def convert(s:String):List[Int] = {if(s.isEmpty) Nil else convert(s.tail):::List((s.head-'0'))}; convert(s)})
  - Note: In the body of a non-primary Scala constructor, ALL you're allowed to do is call the primary constructor. So if we want to call another function, we must actually bundle it as part of the parameter! Looking at the convert function, you'll notice that Scala lets you treat a string as a list of characters. This function also uses the append operator ::: which joins two lists together.
- Also include a constructor that takes no arguments and initializes the list to 0::Nil.
- Implement a + function that adds together two Biguints and returns a new Biguint holding the result. Think back to how you did addition in elementary school, and don't forget the carry. You may use a helper function (in fact I'd recommend it).

3. (5 points) Write a function def findlast(xs:List[Int], x:Int):Int that returns the index of the last time the element x appears in the list xs. Return -1 if the element does not appear in the list. For this function, you MUST use pattern matching, and you may NOT use any built-in list functions (isEmpty, head, tail, map, reduce, etc). I will give 2 points extra credit if you do it without a helper function.

Please name the file you submit for problem 2 HW7-2-Lastname.scala. You can submit your functions for problems 1, 3 in one file called HW7-1-3-Lastname.scala