**Programming HW 8**

**CS 169 Spring 20, Linnell**

**Scala**

**Due Wednesday 5/27 11:59PM**

**Submission instructions at the end**

1. (7 points) Read the blog post here: [https://www.smashingmagazine.com/2014/07/dont-be-scared-of-functional-programming/ (Links to an external site.)](https://www.smashingmagazine.com/2014/07/dont-be-scared-of-functional-programming/" \t "_blank)

 (you only need to read up to “Let’s Get Real”)

Write a 100-150 word summary of this article; you don’t have to rehash the whole running example, but please provide at least three ideas that you found interesting in the article. If you want to read something a little deeper, here’s another blog post you may find interesting: [http://www.ibm.com/developerworks/library/j-ft20/ (Links to an external site.)](http://www.ibm.com/developerworks/library/j-ft20/" \t "_blank)

1. (5 points) Use Call by Name to write a function forLoop in Scala that mimics the behavior of a typical for loop from procedural programming.  forLoop should have two parameter groups.  The first parameter group should take in an integer value i, a condition function that must yield true when applied to i in order for the loop to continue, and an increment function that when applied to i yields the value of i on the next iteration of the loop.  The second parameter group should contain a single parameter of type Unit, the expression to be evaluated each time through the loop.  Write a function call to forLoop that will print out “Hello World” 10 times.

1 point extra credit:  What is the most important shortcoming of the function you’ve written when compared to a for loop in C++?

1. (5 points) Use Call by Name to write Boolean functions and and or that implement the typical binary Boolean operators using short-circuit evaluation; that is, they only evaluate their second parameter if necessary.  Do this without using && or || (looks like & and | are actually bitwise operators, so don’t use those either.  :)

4. Classes (8 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, So the number 1,472 would be stored in the list:  2::7::4::1::Nil.    **(You should be able to do this problem after watching Friday’s videos)**

* For your class, the primary constructor can take in a string; for example 1,472 would be represented as “1472” (this allows us to represent integers with an arbitrary number of digits).  Alternatively, your primary constructor can take in a list of Ints, and you can create a secondary constructor that takes in a string.  If you do this, please see email from me on how to implement the secondary string constructor.
* 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 if you like.

Feel free to use this function in your string constructor:

**def** convert(s:String):List[Int] = {

**if**(s.isEmpty) Nil

**else** {

        convert(s.tail):::List((s.head-'0'))

      }

}

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.

Submission instructions: Upload one file for each problem, named Problem1.txt, Problem2.txt, Problem3.txt, Problem4.txt.  To do this, make a copy of the file, and rename it.