HW5: Scala 1

Due Feb 19 by 11:59pm **Points** 25 **Submitting** a file upload

Throughout this class, you are not allowed to use loops or mutable variables in Scala. If you want to use a feature or function we haven't discussed in class, please ask me first.

1. (5 points) Write a function is_prime that takes in an integer x and returns true if x is prime and false if it is not. To do this, you may write a helper "loop" recursive function that takes **one** parameter. You are welcome to create additional non-recursive helper functions. Please define all helper functions inside is_prime.
2. (5 points) Consider the function add_third

```
def add_third(x:Int):Int = {
  if(x<1) 0
  else x+add_third(x-3)
}
```

You will write a more general version of this function, add_fth, where instead of recursing on x-3, we will use a function f to find the value we will recurse with. To help you get started, here are a couple of questions to answer (you don't need to turn in the answers, just write the function)

Step 1: add_fth will take two parameters, a function f, and an Int x. What should the type of f be?

Step 2: How will we use f in the body of the function?

In addition to the function add_fth, include a function call to add_fth that will have the same behavior as a function call to add_third. To do this you must first define a function to pass as a parameter to add_fth.

3. (10 points) Write a more general version of the sum_f and combine functions from class called apply_combine that applies a function f(taken as a parameter) to the integers from 1 to x(taken as a parameter), then combines those values using another function g(taken as a parameter). So apply_combine takes in three parameters: a function f that takes in an Int and returns an Int; a function g that takes in two Ints and returns an Int; and an Int x . It then returns the result of using the function g to combine the values gained by applying f to 1, 2, …, x. For example, if I define

```
def add(x:Int, y:Int) :Int = x+y
def square(x:Int):Int = x*x
```

then

apply_combine(square, add, 4)

would return 30=16+9+4+1

And if I define

```
def mult(x:Int, y:Int) :Int = x*y
    def identity(x:Int) = x
```

then

apply_combine(identity, mult, 4)

would return 24

4. (5 points) Read the blog post here: **https://www.smashingmagazine.com/2014/07/dont-be-scared-of-functional-programming/ (https://www.smashingmagazine.com/2014/07/dont-be-scared-of-functional-programming/)**

(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. This article uses javascript; you should be able to follow it without any javascript background (one helpful tip: something that looks like this:

```
function(item) {
   return item[propertyName];
 }
```

Is defining an anonymous function.)

Please upload the first three problems as part of the same file. Please name it HW5Yourlastname.scala (or save and submit as a .txt if that's easier to submit) where Yourlastname is your last name. Name the file for problem 4 HW5P4Yourlastname.pdf (or .txt)