Homework 9

Due Mar 15 by 11:59pm **Points** 15 **Submitting** a file upload

1. (10 points) Trace each function call with cbv (for all parameters) and cbn (for all parameters), and count the number of steps needed to do the calculation with each parameter-passing mechanism. Which “wins”?

```
def square(x:Int):Int = x*x

def foo(x:Int, y:Int, z:Int):Int = if(x==y) x*x else z


foo(1+3, 2+2, 5)

foo(1, 1, 6+8*square(3))

foo(1+3, square(2), 4+square(5))

foo(3*2, 12, 6)
```

Please name your file for this problem HW9P1Lastname.pdf

2. (5 points) Use Call by Name to write a function cond that takes in a Boolean and returns: a function that takes in two expressions, and returns the value of the first expression if cond is true, or the value of the second expression if cond is false. Your function should only ever evaluate one of the two expressions sent as parameters to the inner function. For instance if we had the function call below:

```
cond(5<8)(

{

println("execute first expression");

square(2)

},

{

println("execute second expression");

4+3*2
```

})

the value of the function call would be 4 (assuming you have the square function defined as usual) and only "execute first expression" would print out.

I suggest you return a named inner function, and leave off the return type of cond (not required, as it's not recursive).

Please name your file for this problem HW9P2Lastname.scala.

3. Classes (5 points extra credit). Fill in the class and functions below as described in the comments.

```
class Set(f:Int=>Boolean){

//the function f returns true for elements of the set and false for all other numbers

  def contains(elem: Int): Boolean =

      //returns true if elem is in the set and false for all other numbers


  def \/ (t:Set):Set =

  //Returns the union of this set and t.


  def /\ (t:Set):Set =

  //Returns the intersection of this set and t


  def - (t:Set):Set =

  //Returns the difference of this set and t


  def filter(p:Int=>Boolean):Set =

  // Returns a new set that consists of the elements of s that satisfy the predicate.


  def forall(p:Int=>Boolean):Boolean =

  //Returns true if the predicate is true for all elements of this set, and false otherwise.

  // In order to make it possible to implement this function, we will consider a predicate true for all integers if it is true for integers from -1000 to 1000.

}
```

Please name your file for this problem HW9P3Lastname.scala.