Name: Spencer Milbrandt
Partner: Ryan Whitmer
Principles of Programming Languages
Lab 1 Writeup

2. Scala Basics: Binding and Scope. For each of the following uses of names, give the line where that name is bound. Briefly explain your reasoning (in no more than 1-2 sentences).

(a) Consider the following Scala code.

```
1     val pi = 3.14 // definition
2     def circumference (r: Double): Double = { // definition of r
3           val pi = 3.14159 // definition of pi
4           2.0 * pi * r // use of pi and r
5     }
6     def area (r:Double): Double =  // definition or r
7           pi * r * r // use or r
```

The use of pi at line 4 is bound at which line?
pi at line 4 is bound at line 3 because of the scope at which pi was declared in the function. pi is re-declared at line 3 within the circumference definition and therefore is mapped to 3.14159 until the bracket that ends the scope, which in this case is on line 5.

The use of pi at line 7 is bound at which line?
pi at line 7 is bound at line 1. pi at line 3 is defined within the def circumference definition and therefore doesn't affect the val pi definition on line 1. Due to the pi use at line 7, it is bound to the pi at line 1 since the pi value is defined for the entire program unless changed within a certain scope, thus the scope in which pi at line 7 is used in, uses the pi value at line 1.

(b) Consider the following Scala code.

```
1     val x = 3 // definition of x
2     def f(x: Int): Int = // definition of x
3           x match { // use of x
4                 case 0 => 0
5                 case x => { // definition of x
6                       val y = x + 1 // use of x
7                       ({
8                             val x = y + 1
9                             y
10                      } * f(x-1)) // use of x
```

```
11                  }
12            }
13      val y = x + f(x) // use of x
```

The use of x at line 3 is bound at which line?
At line 3, the x is bound by the value x being passed into f() at line 2. At line 3, x is defined as the function's lone parameter.

The use of x at line 6 is bound at which line?
At line 6, the x is still bound by the function's lone parameter being passed in at line 2. Only when x does not equal 0, will line 6 execute, which holds the value that was passed into the function at line 2.

The use of x at line 10 is bound at which line?
At line 10, the x that is being passed into f() is still bound by the lone parameter being passed into the function at line 2.The x declared on line 8 is in a different scope than the reference to x on line 10.

The use of x at line 13 is bound at which line?
At line 13, the x is outside the scope of the function f(), therefore x is actually bound to the value declaration on line 1.

3. Scala Basics: Typing. In the following, I have left off the return type of function g. The body of g is well-typed if we can come up with a valid return type. Is the body of g well-typed?

```
1     def g(x: Int) = {
2           val (a,b) = (1, (x,3))
3           if (x == 0) (b,1) else (b, a + 2)
4     }
```

If so, give the return type of g and explain how you determined this type. For this explanation, first, give the types for the names a and b. Then, explain the body expression using the following format:

e : t because
e_1 : t_1 because
...
e_2 : t_2 because
...

where e_1 and e_2 are subexpressions of e. Stop when you reach values (or names). As an example of the suggested format, consider the `plus` function:

```
def plus(x: Int, y: Int) = x + y
```

Yes, the body expression of `plus` is well-typed with type Int.

x + y: Int because
x: Int
y: Int

If so, give the return type of g and explain how you determined this type?

(a) Yes, the body of g is well-typed with a valid return type of ((Int, Int), Int)
(b) (a,b) : (Int, (Int, Int)) because
a : Int because
1 : Int
b : (Int, Int) because
x : Int
3 : Int