Lab 1 **Kayla Gehring**
CSCI 3155, Spring 2018 Teammate: Tom Eldar

Written Problems

2. Scala Basics: Binding and Scope

For each of the following uses of names, give the where that name is bound. Briefly explain your reasoning.

(a) *The use of pi at line 4 is bound at which line? The use of pi at line 7 is bound at which line?*

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

Line 4 is within the definition of 'circumference' from lines 2-5, and uses the value 'pi' bound at line 3 (val pi = 3.14159).

Line 7 is within the definition of 'area,' which does not bind 'pi' to any value within it, and therefore uses the value 'pi' bound in the global scope at line 1 (val pi = 3.14).

(b) *The use of x at line 3 is bound at which line? The use of x at line 6 is bound at which line? The use of x at line 10 is bound at which line? The use of x at line 13 is bound at which line?*

```
1       val x = 3
2       def f(x: Int): Int =
3       x match {
4         case 0 => 0
5         case x => {
6           val y = x + 1
7           ({
8             val x = y + 1
9             y
10          } * f(x - 1))
11        }
12      }
13    val y = x + f(x)
```

The use of x at line 3 is bound at line 2 (def f(x: Int): Int =), because it's within the scope of the function 'f'.

The use of x at line 6 is bound at line 2, again because it's within the scope of function 'f,' and previous uses of 'x' in the scope are for pattern matching.

The use of x at line 10 is bound at line 2. The binding of x at line 8 is only valid at line 8 and 9, and ends at the { on line 10.

The use of x at line 13 is bound at line 1 (val x = 3) in the global scope. This is because line 13 outside the scope of the function 'f' and all scopes within 'f'.

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     }
```

Yes, the body expression of 'g' is well-typed with a nested tuple of type (Int,(Int, Int)).

```
(b, c):((Int, Int), Int) because
     b:(Int, Int) because
          val (a, b) = (1, (x, 3))
               x:Int
               3:Int
     c:Int because
          1:Int (when x == 0)
          a + 2:Int (when x != 0) because
               val (a, b) = (1, (x, 3))
                   1:Int
               2:Int
```