DS 644: Homework 1

Instructions. Answer the following multiple choice questions by selecting the correct choices.

1. (6 points) **Programming Paradigms**

   (a) Which of the following is *not* an example of a programming paradigm?

   ☐ JavaScript ☐ Declarative ☐ Imperative ☐ Functional ☐ Object-oriented

   (b) Which of the following characteristics are typical of imperative programs.

   ☐ values of variables may change or "mutate" (they are *mutable*)
   ☐ program execution proceeds by carrying out a sequence of instructions
   ☐ functions often have *side-effects*
   ☐ all of the above

   (c) Which of the following characteristics are typical of functional programs.

   ☐ values of variables do not change or "mutate" (they are *immutable*)
   ☐ functions are *referentially transparent*
   ☐ functions do not have *side-effects*
   ☐ all of the above

2. (2 points) A *higher-order function* is a function that

   ☐ can be passed as an argument to other functions

   ☐ can be returned as output by other functions

   ☐ can be called a higher order of times than ordinary, "lower-order" functions

   ☐ accepts a function (or functions) as input or returns a function (or functions) as output.

   ☐ takes a higher order of magnitude of time to return a value than ordinary, "lower-order" functions

3. (2 points) An expression e is called *referentially transparent* provided

   ☐ the value of e, when it is reduced to "normal form," is obvious or "transparent."

   ☐ the values all expressions to which e refers are obvious or "transparent."

   ☐ for all programs p, all occurrences of e in p can be replaced by the result of evaluating e without affecting the meaning of p.

   ☐ none of the above

4. (6 points) **Introduction to Scala, Part I**

   (a) The programming paradigm(s) of Scala is(are) which of these? (select all that apply).

   ☐ assembly   ☐ declarative   ☐ imperative   ☐ functional   ☐ object-oriented

   (b) What is the result of the following program?

```
val x = 0
def f(y: Int) = y + 1
val result = {
  val x = f(3)
  x * x
} + x
```

   ☐ 0   ☐ 16   ☐ 32   ☐ it does not terminate

   (c) Why should we care about writing functions that are "tail-recursive?"

   ☐ Recursion should be carried out on the tail, not the head.

   ☐ Recursion should be carried out on the head, not the tail.

   ☐ Non-tail-recursive functions may exhaust stack memory.

   ☐ Non-tail-recursive functions may exhaust heap memory.

5. (6 points) Consider the following code.

```
def sq(x: Double): Option[Double] =
  if (x < 0) None
  else Some(Math.sqrt(x))

val list = List(-1.0, 4.0, 9.0)
```

   (a) To what does the expression `list.map(sq)` evaluate?

   ☐ `List(2.0, 3.0)`

   ☐ `List(None, Some(2.0), Some(3.0))`

   ☐ `Some(List(2.0, 3.0))`

   ☐ `None`

   ☐ none of the above

   (b) To what does the expression `list.flatMap(sq)` evaluate?

   ☐ `List(2.0, 3.0)`

   ☐ `List(None, Some(2.0), Some(3.0))`

   ☐ `Some(List(i, 2.0, 3.0))`

   ☐ `None`

   ☐ none of the above

6. (4 points) **Introduction to Scala, Part II**. The parts below refer to the function `test(x:Int, y:Int) = x * x`.

(a) For the function call `test(2, 3)`, which evaluation strategy is most efficient (takes the least number of steps)?

☐ call-by-value is more efficient

☐ call-by-name is more efficient

☐ call-by-value and call-by-name require the same number of steps

☐ the program does not terminate

(b) For the function call `test(3 + 4, 8)`, which evaluation strategy is most efficient?

☐ call-by-value is more efficient

☐ call-by-name is more efficient

☐ call-by-value and call-by-name require the same number of steps

☐ the program does not terminate

(c) For the function call `test(7, 2*4)`, which evaluation strategy is most efficient?

☐ call-by-value is more efficient

☐ call-by-name is more efficient

☐ call-by-value and call-by-name require the same number of steps

☐ the program does not terminate

(d) For the function call `test(3+4, 2*4)` which evaluation strategy is most efficient?

☐ call-by-value is more efficient

☐ call-by-name is more efficient

☐ call-by-value and call-by-name require the same number of steps

☐ the program does not terminate