**Review Questions**

* What is Git?
  + Git is a version control software that allows a developer to easily keep track of changes made to their program and to restore earlier version(s), if necessary.
* What about GitHub? How are they related?
  + GitHub is cloud-based repository for Git projects allowing multiple developers to collaborate on a project and see the changes others have made to the code without necessarily having their own source code changed. Changes made in Git on the local machine can be pushed to the GitHub repository allowing others to pull those change to their own local repositories.
* What command moves changes from GitHub to our local repo?
  + Git pull
* What command moves changes from our local repo to GitHub?
  + Git push
* What two commands are used in sequence to save changes to our local repo?
  + Git add and git commit (git commit -a can be used to perform both steps at once
* What is BASH?
  + Stands for Bourne Again Shell
  + One of the most common Unix based shells used to interact with a computer or server from the command line
* What does the PATH environment variable do?
  + The PATH variable tells BASH which directories to search in order to find the binary files for commands in the file structure. This allows the command to be run from the command line simply by typing the command from any location
* What is the JDK and what does it let us do?
  + The JDK is a tool used by Java developers that allow for the compilation of Java code into .class files as well as the execution of these files. JDK contains the JRE.
* What is the JRE and what does it let us do?
  + The JRE is compiler that complies .class files into machine code and then runs them within the JVM. The JRE contains the JVM
* What's the name for the code written in .class files?
  + Bytecode
* How does Scala relate to Java?
  + Scala relates to Java in a number of ways, including the use of the JRE and JVM as well as the use of many Java libraries. It also uses many similar data types as Java
* What is sbt (high level)?
  + The scala build tool. It is a a primary build tool for scala projects and offers several feature including incremental compilation, allowing for faster and less resource intensive compilation
* How does Scala relate to the JRE and JVM?
  + Scala uses JRE and JVM in a similar manner to Java, in that it’s bytecode files are turned into machine code by the JRE and then run on the JVM
* Is Scala statically or dynamically typed?
  + Scala is statically typed i.e. the data type is known at compilation
* Do we need to declare type for our variables in Scala? always?
  + It is not necessary to declare variable types in Scala as Scala is able to infer the data type based on input. Data type can be declare however using the syntax “:” followed by the data type. EX: var x = 44: Int
* What are integer types in Scala?
  + Integers in Scala come in 5 different types:
    - Byte: 8 bits/1 byte, -2^7 through 2^7 -1
    - Short: 16 bits/2 bytes, -2^15 through 2^15 -1
    - Int: 32 bits/ 4 bytes, -2^31 through 2^31 -1
    - Long: 64 bits/8 bytes, -2^63 through 2^63 -1
    - BigInt: any size
* Decimal types?
  + Decimals in Scala come in 3 different types:
    - Float: 4 bytes
    - Double: 8 bytes
    - BigDecimal: any precision
* What is the size of an Int? a Double? (the default integer and decimal types)
  + By default, integers are stored as Ints(4 bytes) unless otherwise specified, whole Decimals are stored as Doubles (8 bytes) unless otherwise specified.
* What is the difference between val and var?
  + Val is an immutable variable, while var is mutable variable
* Why do we make use of programming paradigms like Functional Programming and Object-Oriented Programming?
  + Programming paradigms are used to allow for more consistent structuring of code and thus facilitate collaborate projects that involve multiple developers.
* What are the 4 pillars of OOP: Abstraction, Encapsulation, Inheritance, Polymorphism? (a bit on each)
  + Abstraction is the concept that only the data and methods necessary to use an object are "visible" to the end user. In other words, it is not necessary to understand data structure or methods that an object/class has or how they work in full detail because abstraction allows the use of methods and data structures without needing to look "under the hood"
  + Encapsulation is a concept with respect to protecting the internal workings of an object or class by allowing an object to only be manipulated by itself. Other objects can make a call to a method that an object uses to manipulate itself but cannot directly manipulate the data of another object.
  + Inheritance allows for child objects to "inherit" data structures and methods from parent objects, allowing for reusability of code and thus preventing the need to rewrite code from parent classes in every subsequent child class
  + Polymorphism allows for the code that is inherited to behave differently in different contexts. Two child classes may both inherit the same method from their parent class, however, polymorphism allows the method to function differently in the context of each child class if necessary.
* What are classes? Objects?
  + Classes are blueprints for objects that contain data and methods. For a class to be used, it must be instantiated as an object which is then stored in memory and can be used as needed without affecting the class itself.
* What is the significance of the Object class in Scala?
  + The Object class is the superclass of all other classes and thus all objects in Scala. This means that all objects inherit certain properties from the object class, such as the toString method.
* What does it mean that functions are first class citizens in Scala?
  + The terms “first class citizen” refers to something that cvan be passed as a parameter to a function, returned from a function, or stored as a variable. Scala allows for function to be used in all these ways.
* What is a pure function?
  + A pure function is a function that takes an input and produces and output with no “side effects”, i.e. a mathematical function
* What is a side effect?
  + A side effect occurs when a chunk of code does something other than simply return a value, such as performing I/O, i.e. printing to the console
* What's the difference between an expression and a statement?
  + An expression is used to return a value, whereas a statement is used for it’s side effects, i.e. what it does rather than what it returns.
* Is if-else an expression or a statement in Scala?
  + An if-else is an expression. An if-else is a statement in that it evaluates some form of logic, however it can be used as an expression in order to return a value
* What's the difference between a while loop and a do-while loop?
  + A while loop will only execute while so condition is true, thus it may never execute if the condition is never met. A do-while loop, however, will always execute at least once.
* How does String interpolation work in Scala?
  + String interpolating is used with the syntax (s”Some text ${variable}) allowing for a cleaner way to write Strings and include variable in them.
* How do operators work in Scala? Where is "+" defined?
  + Operators are methods. + sign is defined in the string class and integer class
* What are the ways we could provide default values for a field?
  + Default values for parameters can be defined either in aux constructors or in the primary constructor through the of the “=” operator
* How do we define an auxiliary constructor?
  + An aux constructor is a constructor defined within the body of a class that calls the main constructor but has fewer arguments than the main constructor, thus allowing an object to be instantiated using less parameters than the primary constructor requires and assigning default values as assigned in the aux constructor instead.
* What is an enumeration?
  + An enumeration is a case object that extend a sealed trait for a given object. This solves the problem of using String flags by limiting the number of potential inputs to a class to prevent issues in resolving different spellings, capitalizations, etc. That may result from different developers, i.e. pizza example
* What is a REPL?
  + REPL stands for Read, Eval, Print, Loop and is essentially a command line interpreter that allows a developer to enter lines directly into the command line. This can be useful when a developer wishes to test various bits of code quickly without altering the source code and having the recompile in between every edit. Very useful for debugging code
* How does the Scala REPL store the results of evaluated expressions?
  + The Scala REPL store the results of evaluated expression as res# where # = an integer starting at 1 and ascending in order
* What is a higher order function?
  + Higher order functions take other functions as parameters or return a function as a result.
* What is function composition?
* Why might we want to write an application using pure functions instead of impure functions?
* Why might we want mutable state (OOP) in our application? Why might we want immutable state (FP)?
  + This question is open-ended, don't focus on finding the "right" answer
  + A mutable list for example would be a List that contains 1, 2, 3, 4. If I append that list to add 5 to it, it becomes mutable because the state of the list was manipulated. An immutable list using the same example would create a NEW list that appends the number 5.
* What are some examples of side effects?
* What is a Lambda?
* How can we write Lambdas in Scala?
* What does map do?
* What does filter do?
* What does reduce do?
* What does it mean that a function returns Unit in Scala?
* What is recursion?
* What do we need to include in any effective recursive function?
* What is the Stack? What is stored on the Stack?
* What is the Heap? What is stored on the Heap?
* What is garbage collection?
* When is an object on the Heap eligible for garbage collection?
* How do methods executing on the stack interact with objects on the heap?
* Know the basics of the following Scala Collections: (mutable? indexed? when to use it?)
  + List
  + Set
  + Map
  + ArrayBuffer
  + Vector
* What is a generic?
  + the phrase "compile time type safety" is useful in this answer
  + A generic class is a class which takes a type as a parameter. They are particularly useful for collection classes. During compilation time, this helps to avoid exceptions.
* What is an Option?
  + Can be used to represent optional values. Instances of Option are either an Instance of “Some” or the object “None”.
* What is found inside an empty Option?
  + The object “None”.
* How do we write an Option that contains a value?
* What are Exceptions?
* What are Errors?
* What is Throwable?
* How would we write code that handles a thrown Exception?
* How do we write code to throw an Exception?
  + Using a match – case
* What does the src folder contain in a sbt project?
  + It contains the source code that are saved in “.scala” files. It also contains the main class that is last in the stack considering LIFO.
* the target folder?
* What is a Trait?
  + Traits are similar to a Java interface. It encapsulates methods and fields.
* What is a case class?
  + Case classes are classes that are given syntactic sugar that are useful when defining immutable classes. They are used for structuring data in FP-style code. Fields are val by default, accessor methods are generated but vals cannot be changed.
* What methods get generated when we declare a case class?
* What does the apply method do?
* What is a Thread?
* What is a Future?
* In what situations might we want to use multiple threads? (just some)
* How can we do something with the result of a Future?
* How does a Future return a value or exception?
* How does the onComplete callback function work?