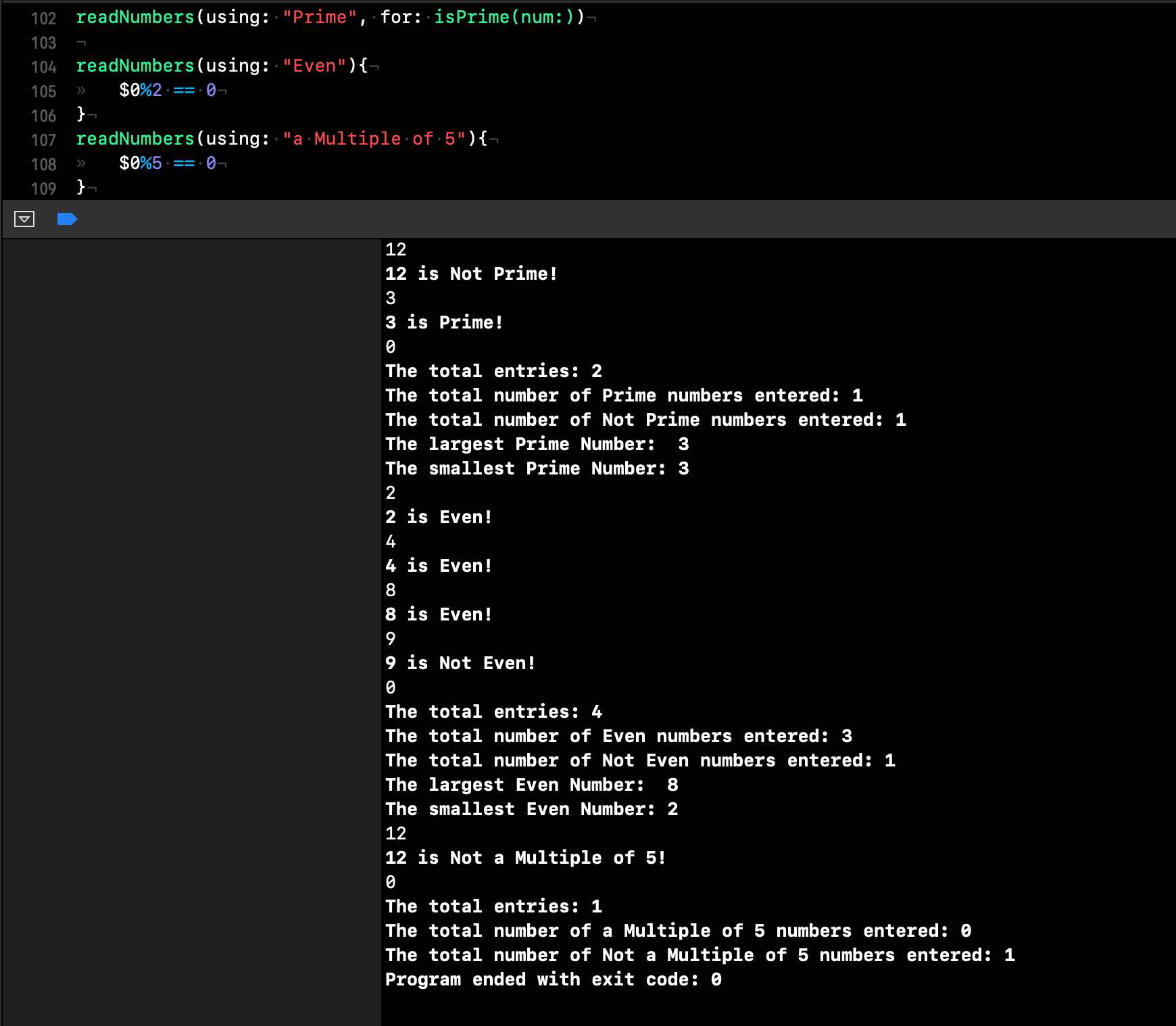
**Objective:** To apply what we’ve learned about functional programming and closures, their uses, and syntax to a problem we’ve already solved.

**Summary:** *By this point, you’ve hopefully figured out some solution to the primeFinder project (if not make sure you come to see me in office hours!). While primeFinder was a good project to get you started with Swifty programming, we can make it better. We are going to apply what we’ve discussed about Functional Programming to the primeFinder idea and make it into a versatile program.*

**Directions:**

1. *Either open up primeFinder or create a new project and copy the code from primeFinder’s main.swift file into the new project’s main file*
2. *To make this more functional and therefore more flexible, you will have to separate the “general process stuff” from the primeness tests.*
   1. *Create two functions to hold the code that was previously in primeFinder:*
      1. **func readNumbers(using label: String, for process: (Int)->Bool ){...}**
      2. **func isPrime(\_ num: Int) -> Bool{...}**
         1. **Note the … is because I’m not giving you the body of the functions, you have to figure that part out**
   2. ***readNumbers*** *should handle the input from the user, checking entry validity, conversions,calling* ***process,*** *tracking valid & invalid entries, tracking largest/smallest valid entries (if/any), and printing*
      * 1. *Normally this would probably be broken up even further into smaller functions but that is not the focus here, so don’t worry about it*
        2. *Use the* ***label*** *for the printouts. So instead of “Prime”/“Not Prime” use “\(label)/Not \(label)”*
        3. ***process*** *is a function/closure that will be used as the validity test.*
        4. *You may want to change whatever numPrimes/nonPrimes variables you used previously to valid/invalid so it makes more sense when reading the program*
   3. ***isPrime*** *should be checking primeness and returning a boolean value to be used in* ***readNumbers***
      * 1. *Its checking the validity of possibly prime numbers*
3. *Below the new function definitions:*
   1. *Call the readNumbers function and pass an appropriate label and the isPrime function pointer like this:*
      1. **readNumbers(using: "Prime", for: isPrime)**
4. Now that readNumbers has been generalized for many purposes,demonstrate this by calling the function a few more times with a couple different labels and closures, using ***trailing closure syntax*** as discussed in lecture
   1. Make the labels appropriate for the test
      * 1. Some test labels/ides “Multiple of 5”, “Even”, “Odd”, “
   2. Some examples of my Code running are below:



**(Optional) BONUS CHALLENGE:** This can be made better. Not every number series requires only Integers. You can separate the integer conversion into an isInt Check that returns a boolean.

**func isInt(entry: String) -> Bool**

*Change* ***isPrime*** *to accept a String instead of Int*

*call the function* ***isInt*** *inside* ***isPrime*** *to ensure the number being checked is an Int (non Ints are not Prime by default) (you can also have readNumbers call isInt directly!)*

*Change the process closure’s signature to accept a* ***String.***

*You also probably to need to perform some minor code refactoring in* ***readNumbers***

***Submission:*** *a pdf, jpg, jpeg, or png file that demonstrates your code running with some of the closures suggested (or your own)* ***AND*** *the project source code (****main.swift*** *file****)*** *uploaded to the appropriate canvas site (you’ll have to download it from Apporto when finished)*