Introduction to Swift

Lesson 1 - Individual Worksheet

Understanding Types

Write the type of the constant written in the "Type" column. If there's an error with the code, instead write "Error: " and what the error is in the column.

Code	Type
`let a: Int = 5`	
`let b = 8`	
`let c = −6.7`	
`let d: Double = 5`	
`let e = "Catz!"`	
`let f: Int = 7.8`	
`let g = "5"`	
`let h: Double = "98.7"`	
`let i = 9 + 7`	
`let j = 9.8 + 3`	
`let k: Int = 9.5 − 0.5`	

Writing Constants Clearly

Read the prompts below and write corresponding constants for the problems.

Make sure the names clearly represent what values they hold for the context. The values do not have to be the exact values in the context, but they should make sense type-wise (distances should be Ints or Doubles, names should be Strings).

As an example, the first problem is done for you.

1. You want to write a program to test if the number 600851475143 is prime. Write **one** constant which stores the number that you want to check is prime.

   ```
   let wantToCheckPrime = 600851475143
   ```

2. Suppose you want to make an iPhone app that tells you when to leave for school. It would need to keep track of: the time your first class starts and how long it takes you to travel to school. Write **two** constants to model this problem.

3. Suppose you create a program which tells you what to wear based on the weather. In the morning you input to it the predicted high and low temperatures, the chance of rain, and the average wind speed. The program stores these values in constants, and then suggests something to wear. Write **four** constants to store these values.

4. Voting in some places of the United States is done with electronic voting machines. Suppose there are two political candidates who are running for office, and you are writing software to control electronic voting machines. Specifically, in one part of the software you need to use constants to keep track of the candidates' names, and how many votes each candidate has. Write **four** constants to do so.

Evaluating Code – Compilation Errors

Below is some Swift code. Some lines may produce compiler errors.

First mark all of the lines which will produce compiler errors with asterisks.

Then, please rewrite **any** combination of all of the available lines of code, so that the entire snippet compiles. The line numbers on the left should help you.

In addition, try to maintain the original intent of the code.

```
1>   // We compute what angle to fire a cannon at to hit a target.

2>   // Given parameters:

3>   let distanceToTarget = 800 // The target is 800 meters away

4>

5>   let cannonMuzzleSpeed = 524.256 // The shot leaves at about
          524 m/s

6>

7>

8>   // Computation: First we compute the intermediate value

9>   let distanceSin: Int = 9.8 * distanceToTarget /
          (cannonMuzzleSpeed * cannonMuzzleSpeed)

10>

11> // Then we compute the angle to fire the cannon at

12> let angle = asin(distanceSin) / 2

13>

14> // Also, we compute how long it will take to hit the target

15> let horizontalSpeed = cos(angle) * cannonMuzzleSpeed

16>

17> let timeOfFlight = distanceToTarget / horizontalSpeed
```

Evaluating Print Statements

Below is some Swift code. It compiles fine.

Below it, please write what you think the code will print. If you haven't specifically learned what some code means, that's fine, you should still try to guess what it will do!

```
1>  let fruit1 = "banana"

2>  let fruit2 = "apple"

3>  let fruit3 = "🍑"

4>

5>  let fruit1_num = 1

6>  let fruit2_num = 12.3

7>  let fruit3_num = 6

8>

9>  let name = "Helen"

10>

11> print("\(name), please go to the store for me.")

12> print("I need \(fruit1_num) \(fruit1)s, \(fruit2_num)
        \(fruit2)s, and \(fruit3_num) \(fruit3)s.")
```