Lab 2: Exploring Arrays, Dictionaries, and Structs

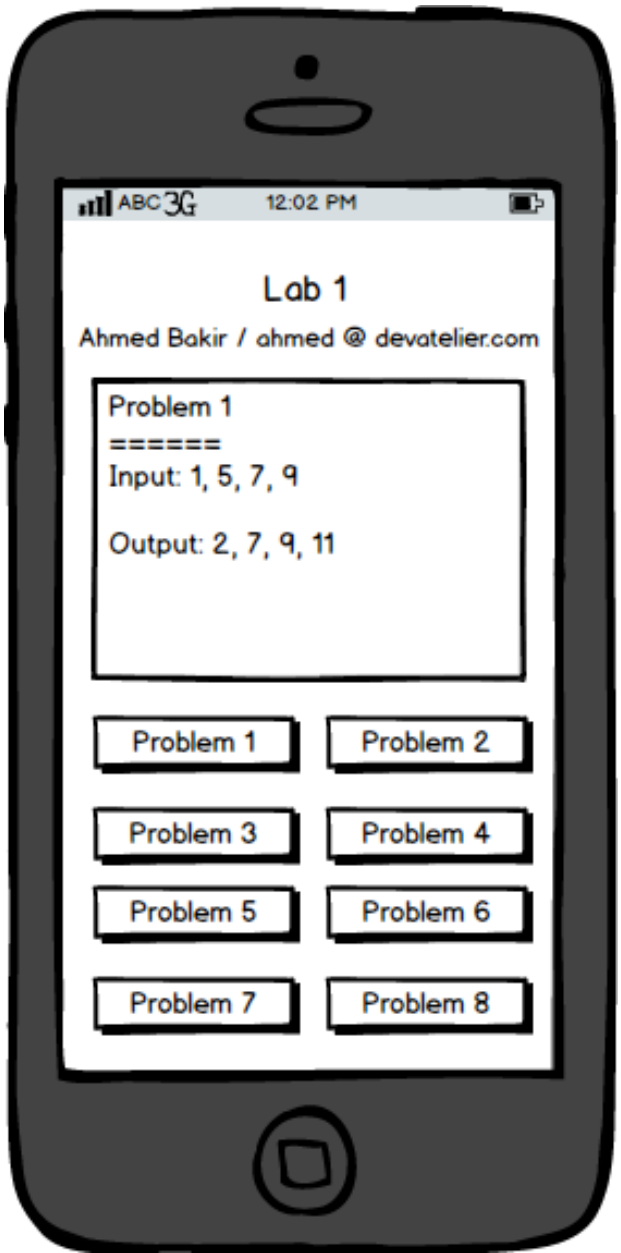
Description

In this lab, you will write a series of functions that explore different operations on structs, arrays, and dictionaries.

Instructions

Create a project named “Lab2”, use your name .com as the company identifier. Or use a domain name you own if applicable.

I will test your solutions using the iPhone 7 Simulator on XCode 8 / OS X 10.11. Implement the user interface indicated below.



As with Lab 1, retain the general look and feel of the user interface. You will not be graded on shadows or other effects, but you are expected to make elements retain the relative sizes and proportions and indicated by the mockup.

Your output should be displayed in a **UITextField**. Text fields allow you to display multiple lines of text without additional instructions in code.

Make a button that corresponds to each problem #. In the button handler, execute the function you are asked to write in the problem.

Ex) Write a function that takes an array of strings and returns that array in reverse order

```
@IBAction func firstButtonPressed {
        var inputArray = [“String 1”, “String 2”, “String 3”]
        var reversed Array = reverseArray(inputArray)
}

fun printArray(inputArray: [String]) -> [String] {
        //implementation goes here
}
```

Display the following after each function executes:

- Line 1: Problem Number (Ex, “Problem 1“)
- Line 2: Five equal-signs (Ex, “=====”)
- Lines 3 and 4: Inputs
  - Use Square brackets to indicate arrays (Ex, Input: [4, 1, 5, 2, 3])
  - Use JSON notation to indicate dictionaries (Ex, Input : {“Key 1”: “Value 1”, “Key 2”: “Value 2”. “Key 3”: “Value 3” }
  - Display an empty line on line 4 if there is only one input
- Line 5: Output
  - Use Square brackets to indicate arrays (Ex, Output: [4, 1, 5, 2, 3])
  - Use JSON notation to indicate dictionaries (Ex, Output : {“Key 1”: “Value 1”, “Key 2”: “Value 2”. “Key 3”: “Value 3” }

UITextField uses an **NSString** for its text input. Use the NSString**(stringWithFormat:) convenience constructor**

Reset the contents of the text field each time a button is pressed. Ex) If you click on the button for Problem 2, it will clear the output from Problem 1 before displaying the output for Problem 2.

Problem 1:
Write a function that takes an array of strings and returns an array that contains the same string in reversed order.
Input: [“String 1”, “String 2”, “String 3“]
Output: [“String 3“, “String 2”, “String 1”]

Problem 2:
Write a function that takes an array of unsorted integer values and returns them in ascending order.
Input: [4, 1, 5, 2, 3]
Output: [1, 2, 3, 4, 5]

Problem 3:
Write a function that takes two arrays of equal length, containing number values, and returns an array of number values containing the sum. The first array contains integer values and the second array contains float values. The result should contain float values. When printing the result, display two decimal places for each item.
Input 1: [4, 1, 5, 2, 3]
Input 2: [-2.06, 2.0, 3.1234, 1.2, -1.2]
Output: [1.94, 3.00, 5.12, 3.20 , 2.80]

Problem 4:
Write a function that takes an array as an input and returns an array containing all but the last two values of the input array. If the input array is less than 2 values long, return nil.
Input: [4, 1, 5, 2, 3]
Output: [4, 1, 5]

Problem 5:
Write a function that takes two arrays of equal length and returns a dictionary. The first array contains the keys and the second array contains values.
Input 1: [“Name”, “Hair Color”, “Eye Color”]
Input 2: [“John”, “Black”, “Brown”]
Output: {“Name” : “John”, “Hair Color” : “Black”, “Eye Color”: “Brown”}

Problem 6:
Create a function that takes an dictionary as input and returns the number of keys as an integer.
Input: {“Name” : “John”, “Hair Color” : “Black”, “Eye Color”: “Brown”}
Output: 3

Problem 7:
Create a function that takes a dictionary and a key (as a string) as input and returns the same dictionary, minus the indicated key-value pair
Input 1: {“Name” : “John”, “Hair Color” : “Black”, “Eye Color”: “Brown”}
Input 2: “Hair Color”
Output: {“Name” : “John”, “Eye Color”: “Brown”}

Problem 8:
Create a struct named Human to represent a person’s name, hair color, and eye color. Create a function that takes a dictionary containing the key-value pairs for a human, and returns a variable of type Human, initialized with all the corresponding values from the dictionary. Use the variable’s name when displaying the output
Input: {“Name” : “John”, “Hair Color” : “Black”, “Eye Color”: “Brown”}
Output: john.name = “John“, john.hairColor = “Black”, john.eyeColor = “Brown”

Grading

You will be graded on:
- Project compatibility - 20%
  - Does it run and compile on XCode 8 / OS X 10.11
  - Does it look like the mockup when I run it on the iPhone 7 simulator?
  - Does it run without crashing when I start it up?
- Basic requirements - 70%
  - Do the buttons change the text in the text field?
  - Do the methods produce the correct output in the text field?
  - Do the buttons call the right methods?
- Theming + Code Standards - 10%
  - Do the methods use the correct types for return values and parameters?
  - Did you follow project naming standards?
  - Do you have all your definitions and declarations in the right places?
  - Did you follow **dot** syntax for sending messages?