TY CSE AY-2023-24 Sem-II

iOS Lab (6CS381)

Assignment No 9

Due date- 26/03/2024

Exercise - Arrays

1. Assume you are an event coordinator for a community charity event and are keeping a list of who has registered. Create a variable `registrationList` that will hold strings. It should be empty after initialization.

2. Your friend Sara is the first to register for the event. Add her name to `registrationList` using the `append(_:)` method. Print the contents of the collection.

3. Add four additional names into the array using the `+=` operator. All of the names should be added in one step. Print the contents of the collection.

4. Use the `insert(_:at:)` method to add `Charlie` into the array as the second element. Print the contents of the collection.

5. Somebody had a conflict and decided to transfer registration to someone else. Use array subscripting to change the sixth element to `Rebecca`. Print the contents of the collection.

6. Call `removeLast()` on `registrationList`. If done correctly, this should remove `Rebecca` from the collection. Store the result of `removeLast()` into a new constant `deletedItem`, then print `deletedItem`.

Activity Challenge

>These exercises reinforce Swift concepts in the context of a fitness tracking app.

7. Your fitness tracking app shows users a list of possible challenges, grouped by activity type (i.e. walking challenges, running challenges, calisthenics challenges, weightlifting challenges, etc.) A challenge could be as simple as "Walk 3 miles a day" or as intense as "Run 5 times a week."

Using arrays of type `String`, create at least two lists, one for walking challenges, and one for running challenges. Each should have at least two challenges and should be initialized using an array literal. Feel free to create more lists for different activities.

8. In your app you want to show all of these lists on the same screen grouped into sections. Create a `challenges` array that holds each of the lists you have created (it will be an array of arrays). Using `challenges`, print the first element in the second challenge list.

9. All of the challenges will reset at the end of the month. Use the `removeAll` to remove everything from `challenges`. Print `challenges`.

10. Create a new array of type `String` that will represent challenges a user has committed to instead of available challenges. It can be an empty array or have a few items in it.

11. Write an if statement that will use `isEmpty` to check if there is anything in the array. If there is not, print a statement asking the user to commit to a challenge. Add an else-if statement that will print "The challenge you have chosen is <INSERT CHOSEN CHALLENGE>" if the array count is exactly 1. Then add an else statement that will print "You have chosen multiple challenges."

Exercise - Dictionaries

12. Create a variable `[String: Int]` dictionary that can be used to look up the number of days in a particular month. Use a dictionary literal to initialize it with January, February, and March. January contains 31 days, February has 28, and March has 31. Print the dictionary.

13. Using subscripting syntax to add April to the collection with a value of 30. Print the dictionary.

14. It's a leap year! Update the number of days in February to 29 using the `updateValue(_:, forKey:)` method. Print the dictionary.

15. Use if-let syntax to retrieve the number of days under "January." If the value is there, print "January has 31 days", where 31 is the value retrieved from the dictionary.

16. Given the following arrays, create a new [String : [String]] dictionary. `shapesArray` should use the key "Shapes" and `colorsArray` should use the key "Colors." Print the resulting dictionary.

17. Print the last element of `colorsArray`, accessing it through the dictionary you've created. You'll have to use if-let syntax or the force unwrap operator to unwrap what is returned from the dictionary before you can access an element of the array.

Pacing

>These exercises reinforce Swift concepts in the context of a fitness tracking app.

18. In previous app exercises you've written code to help users with run pacing. You decide that you could use a dictionary to let users store different paces that they regularly run at or do interval training with.

Create a dictionary `paces` of type [String: Double] and assign it a dictionary literal with "Easy", "Medium", and "Fast" keys corresponding to values of 10.0, 8.0, and 6.0. These numbers correspond to mile pace in minutes. Print the dictionary.

19. Add a new key/value pair to the dictionary. The key should be "Sprint" and the value should be 4.0. Print the dictionary.

20. Imagine the user in question gets faster over time and decides to update his/her pacing on runs. Update the values of "Medium" and "Fast" to 7.5 and 5.8, respectively. Print the dictionary.

21. Imagine the user in question decides not to store "Sprint" as one his/her regular paces. Remove "Sprint" from the dictionary. Print the dictionary.

22. When a user chooses a pace, you want the app to print a statement stating that it will keep him/her on pace. Imagine a user chooses "Medium." Accessing the value from the dictionary, print a statement saying "Okay! I'll keep you at a <INSERT PACE VALUE HERE> minute

mile pace."

Exercise - For-In Loops

23. Create a for-in loop that loops through values 1 to 100, and prints each of the values.

24. Create a for-in loop that loops through each of the characters in the `alphabet` string below, and prints each of the values alongside the index.

25. Create a `[String: String]` dictionary, where the keys are names of states and the values are their capitals. Include at least three key/value pairs in your collection, then use a for-in loop to iterate over the pairs and print out the keys and values in a sentence.

Movements

>These exercises reinforce Swift concepts in the context of a fitness tracking app.

26. Suppose your app contains a list of different movements that can be tracked. You want to display each item in the list to the user. Use a for-in loop to loop through `movements` below and print each movement.

let movements: [String] = ["Walking", "Running", "Swimming", "Cycling", "Skiing", "Climbing"]

27. Now suppose your app uses a dictionary to keep track of your average heart rate during each of the movements in `movements`. The keys correspond to the movements listed above, and the values correspond to the average heart rate that your fitness tracker has monitored during the given movement. Loop through `movementHeartRates` below, printing statements telling the user his/her average heart rate during each exercise.

var movementHeartRates: [String: Int] = ["Walking": 85, "Running": 120, "Swimming": 130, "Cycling": 128, "Skiing": 114, "Climbing": 129]

Exercise - While Loops

28. Create a while loop that simulates rolling a 6-sided dice repeatedly until a 1 is rolled. After each roll, print the value. (Hint: use `Int.random(in: 1...6)` to generate a random number between 1 and 6).

(Note: Use ‘import Foundation’ library)

Running Cadence

>These exercises reinforce Swift concepts in the context of a fitness tracking app.

29. You may want your fitness tracking app to help runners track and improve their cadence. Running cadence is the number of steps a runner takes in a minute. To help with this, you decide to let the user input a cadence, after which your app will play a sound at each interval they need to take another step.

For this exercise, you'll simulate a "test run" of the cadence feature of your app. Use a while loop to print "Take a step" to the console 10 times. Once you've successfully printed "Take a

step" to the console 10 times, add the following code to the end of your while loop: `Thread.sleep(forTimeInterval: 60/cadence)`. This will put a pause between each itertion of the while loop so that the print statements actually occur at the proper cadence.

(Note: Use ‘import Foundation’ library)

30. Recreate the above cadence example using a repeat-while loop.

Exercise - Control Transfer Statements

31. Create a for-in loop that will loop through `alphabet`. Inside the loop, print every other letter by continuing to the next iteration if you are on a letter you do not wish to print. (Hint: You can use the `isMultiple(of:)` method on `Int` to only print even indexed characters).

32. Create a `[String: String]` dictionary where the keys are names of states and the values are their capitals. Include at least three key/value pairs in your collection, with one of them being your home state. Now loop through this dictionary again, printing out the keys and values in a sentence, but add an if statement that will check if the current iteration is your home state. If it is, print("I found my home!") and break out of the loop.

Finding Movements

>These exercises reinforce Swift concepts in the context of a fitness tracking app.

33. You decide you want your app's users to be able to put in a heart rate range they would like to hit, and then you want the app to suggest movements where historically the user has reached that heart rate range. The dictionary `movementHeartRates` below contains keys corresponding to the movements that the app has tracked, and values corresponding to the average heart rate of the user that your fitness tracker has monitored historically during the given movement.

Loop through `movementHeartRates` below and if the heart rate doesn't fall between `lowHR` and `highHR`, continue to the next movement and heart rate. Otherwise, print "You could go <INSERT MOVEMENT HERE>."

****