TY CSE AY-2022-23 Sem-II

Sub: iOS Lab (6CS381)

Assignment No 6 **Due date- 19/02/2024**

Exercise - Create Functions

1. Write a function called `introduceMyself` that prints a brief introduction of yourself. Call the functionand observe the printout.

2. Write a function called `magicEightBall` that generates a random number and then uses either a switch statement or if-else-if statements to print different responses based on the random number generated. `let randomNum = Int.random(in: 0...4)` will generate a random number from 0 to 4, after which you can print different phrases corresponding to the number generated. Call the function multipletimes and observe the different printouts.

import Foundation

App Exercise - A Functioning App

3. As you may have guessed, functions are key to making your app work. For example, in every exercisedealing with step count until now, you have simply assigned a number of steps to a `steps` variable. Thisisn't very realistic seeing as the number of steps you take increments one at a time and continues changing throughout the day.

A reoccurring process like this is a perfect candidate for a function. Write a function called `incrementSteps` after the declaration of `steps` below that will increment `steps` by one and then printits value. Call the function multiple times and observe the printouts.

var steps = 0

4. Similarly, if you want to regularly provide progress updates to your user, you can put your control flow statements that check on progress into a function. Write a function called `progressUpdate` afterthe declaration of `goal` below. The function should print "You're off to a good start." if `steps` is less than 10% of `goal`, "You're almost halfway there!" if `steps` is less than half of `goal`, "You're over halfway there!" if `steps` is less than 90% of `goal`, "You're almost there!" if `steps` is less than `goal`, and "You beat your goal!" otherwise. Call the function and observe the printout.

let goal = 10000

Exercise - Parameters and Argument Labels

5. Write a new introduction function called `introduction`. It should take two `String` parameters, `name` and `home`, and one `Int` parameter, `age`. The function should print a brief introduction. I.e. if"Mary," "California," and 32 were passed into the function, it might print "Mary, 32, is from California."Call the function and observe the printout.

6. Write a function called `almostAddition` that takes two `Int` arguments. The first argument should notrequire an argument label. The function should add the two arguments together, subtract 2, then print the result. Call the function and observe the printout.

7. Write a function called `multiply` that takes two `Double` arguments. The function should multiply thetwo arguments and print the result. The first argument should not require a label, and the second argument should have an external label, `by`, that differs from the internal label. Call the function and observe the printout.

App Exercise - Progress Updates

8. In many cases you want to provide input to a function. For example, the progress function you wrotein the Functioning App exercise might be located in an area of your project that doesn't have access to the value of `steps` and `goal`. In that case, whenever you called the function, you would need to provide it with the number of steps that have been taken and the goal for the day so it can print the correct progress statement.

Rewrite the function `progressUpdate`, only this time give it two parameters of type `Int` called `steps` and `goal`, respectively. Like before, it should print "You're off to a good start." if steps is less than 10% of goal, "You're almost halfway there!" if steps is less than half of goal, "You're over halfway there!" if steps is less than 90% of goal, "You're almost there!" if steps is less than goal, and "You beat your goal!"otherwise. Call the function and observe the printout.

Call the function a number of times, passing in different values of `steps` and `goal`. Observe the printouts and make sure what is printed to the console is what you would expect for the parameterspasssed in.

9. Your fitness tracking app is going to help runners stay on pace to reach their goals. Write a function called pacing that takes four `Double` parameters called `currentDistance`, `totalDistance`,
`currentTime`, and `goalTime`. Your function should calculate whether or not the user is on pace to hitor beat `goalTime`. If yes, print "Keep it up!", otherwise print "You've got to push it just a bit harder!"

Exercise - Return Values

10. Write a function called `greeting` that takes a `String` argument called name, and returns a `String`that greets the name that was passed into the function. I.e. if you pass in "Dan" the return

value mightbe "Hi, Dan! How are you?" Use the function and print the result.

11. Write a function that takes two `Int` arguments, and returns an `Int`. The function should multiplythe two arguments, add 2, then return the result. Use the function and print the result.

App Exercise - Separating Functions

12. One principle that can help in debugging and maintaining code is abstraction. For example, in your fitness tracking app some of your existing functions have been written to both perform a calculation andprint a message. But it's very possible that you'll decide to change either the calculation or the message in the future. It will be easier to go back and change this if you separate the calculation from the message.

As an example, write a function that only does a portion of what your previous `pacing` function did.This function will be called `calculatePace`. It should take three `Double` arguments called `currentDistance`, `totalDistance`, and `currentTime`, and should return a `Double` that will representthe time at which the user will finish the run based on the user's current distance and time. call the function and print the return value.

13. Now write a function called `pacing` that takes four `Double` arguments called `currentDistance`, `totalDistance`, `currentTime`, and `goalTime`. The function should also return a `String`, which will be the message to show the user. The function should call `calculatePace`, passing in the appropriate values, and capture the return value. The function should then compare the returned value to
`goalTime` and if the user is on pace return "Keep it up!", and return "You've got to push it just a bit harder!" otherwise. Call the function and print the return value.

****