Department of Computer Science and Engineering

Sub: iOS application development using swift

Assignment on Optionals and Enumerations

1. Imagine you have an app that asks the user to enter his/her age using the keyboard. When your app allows a user to input text, what is captured for you is given as a `String`. However, you want to store this information as an `Int`. Is it possible for the user to make a mistake and for the input to not match the type you want to store?

2. Declare a constant `userInputAge` of type `String` and assign it "34e" to simulate a typo while typing age. Then declare a constant `userAge` of type `Int` and set its value using the `Int` initializer that takes an instance of `String` as input. Pass in `userInputAge` as the argument for the initializer. What error do you get?

3. Go back and change the type of `userAge` to `Int?`, and print the value of `userAge`. Why is `userAge`'s value `nil`? Provide your answer in a comment or print statement below.

4. Now go back and fix the typo on the value of `userInputAge`. Is there anything about the value printed that seems off? Print `userAge` again, but this time unwrap `userAge` using the force unwrap operator.

5. Now use optional binding to unwrap `userAge`. If `userAge` has a value, print it to the console.

   ## App Exercise - Finding a Heart Rate

6. Many APIs that give you information gathered by the hardware return optionals. For example, an API for working with a heart rate monitor may give you `nil` if the heart rate monitor is adjusted poorly and cannot properly read the user's heart rate. Declare a variable `heartRate` of type `Int?` and set it to `nil`. Print the value.

7. In this example, if the user fixes the positioning of the heart rate monitor, the app may get a proper heart rate reading. Below, update the value of `heartRate` to 74. Print the value.

8. As you've done in other app exercises, create a variable `hrAverage` of type `Int` and use the values stored below and the value of `heartRate` to calculate an average heart rate.

   ```
   let oldHR1 = 80
   let oldHR2 = 76
   let oldHR3 = 79
   let oldHR4 = 70
   ```

9. If you didn't unwrap the value of `heartRate`, you've probably noticed that you cannot

perform mathematical operations on an optional value. You will first need to unwrap `heartRate`. Safely unwrap the value of `heartRate` using optional binding. If it has a value, calculate the average heart rate using that value and the older heart rates stored above. If it doesn't have a value, calculate the average heart rate using only the older heart rates. In each case, print the value of `hrAverage`.

10. Define a `Suit` enum with four possible cases: `clubs`, `spades`, `diamonds`, and `hearts`.
    */

11. Imagine you are being shown a card trick and have to draw a card and remember the suit. Create a variable instance of `Suit` called `cardInHand` and assign it to the `hearts` case. Print out the instance.

12. Now imagine you have to put back the card you drew and draw a different card. Update the variable to be a spade instead of a heart.

13. Imagine you are writing an app that will display a fun fortune (i.e. something like "You will soon find what you seek.") based on cards drawn. Write a function called `getFortune(cardSuit:)` that takes a parameter of type `Suit`. Inside the body of the function, write a switch statement based on the value of `cardSuit`. Print a different fortune for each `Suit` value. Call the function a few times, passing in different values for `cardSuit` each time.

14. Create a `Card` struct below. It should have two properties, one for `suit` of type `Suit` and another for `value` of type `Int`.

15. How many values can playing cards have? How many values can `Int` be? It would be safer to have an enum for the card's value as well. Inside the struct above, create an enum for `Value`. It should have cases for `ace`, `two`, `three`, `four`, `five`, `six`, `seven`, `eight`, `nine`, `ten`, `jack`, `queen`, `king`. Change the type of `value` from `Int` to `Value`. Initialize two `Card` objects and print a statement for each that details the card's value and suit.

****