Programación de aplicaciones para dispositivos iOS

Realiza los siguientes ejercicios codificando playgrounds en Swift.

1) **Codifica 3 diccionarios en Swift, que contengan información que consideres pertinente.**

2) **Dada la siguiente struct:**

```
struct Person {
    var name: String
    var age: Int

    func compareAge(_ p: Person) -> String {
        // Write code here!

    }
}
```

Escribe en la estructura Person el código necesario que permita devolver cómo se compara la edad de otra persona. Dadas las instancias p1, p2 y p3, que se inicializarán con los atributos name y age, codifíca el método de manera que devuelva una sentencia en el siguiente formato:

```
p1 = Person("Samuel", 24)
p2 = Person("Joel", 36)
p3 = Person("Lily", 24)
```

```
p1.compareAge(p2) → "Joel is older than me."

p2.compareAge(p1) → "Samuel is younger than me."

p1.compareAge(p3) → "Lily is the same age as me."
```

3) **Los cajeros automáticos permiten códigos PIN de 4 o 6 dígitos y los códigos PIN no pueden contener nada más que exactamente 4 dígitos o exactamente 6 dígitos. Crea una función en Swift que tome una cadena y devuelva verdadero si el PIN es válido y falso si no lo es.**

Ejemplo:

```
validatePIN("1234") → true
```

```
validatePIN("12345") → false

validatePIN("a234") → false

validatePIN("") → false
```

4) **Crea una función en Swift que tome un arreglo de enteros no negativos y cadenas que devuelva un nuevo arreglo sin las cadenas.**

Ejemplo:

```
filterArray([1, 2, "a", "b"]) → [1, 2]

filterArray([1, "a", "b", 0, 15]) → [1, 0, 15]

filterArray([1, 2, "aasf", "1", "123", 123]) → [1, 2, 123]
```

El cero es un número entero no negativo.

La matriz dada solo tiene números enteros y cadenas.

Los números de la matriz no deben repetirse.

Debe mantenerse el orden original.

Sugerencia: Tu arreglo puede declararse así:

```
var array: [Any] = [0, 1, 2, “a”, “b”]
```

5) **Crea una clase en Swift que modele un Vehículo. Esta clase deberá contener los atributos nombre, modelo, kilometraje y precio, así como el método conducir, que desplegará un mensaje indicando el nombre del vehículo que se está conduciendo.**

class Vehicle {

// Write code here!!

}

Crea dos subclases de la clase Vehículo, denominadas Compacto y Camión. Para cada subclase:

- **Especifica 3 atributos más que los diferencien de la clase Vehículo. (Definidos por ti)**
- **Crea un método particular para cada subclase (Definidos por ti). Cada método debe desplegar información del Vehículo. Cada subclase heredará el método conducir de la clase principal.**

Ejemplo:

//Para un vehículo normal

`Vehículo1.seeModel()` → `I'm model 2019`