Practica 2:
Tipos de datos

Ejercicio 1

El código siguiente corresponde al lenguaje de programación Objective-C.

```
// Animal.h
@interface Animal : NSObject
  @property NSString* name;
  -(void) makeSound;
@end

// Animal.m
@implementation Animal
  -(void) makeSound {
    NSLog(@"The animal makes a sound");
  }
@end

// Cat.h
@interface Cat : Animal
  -(void) sleep;
@end

// Cat.m
@implementation Cat
  -(void) makeSound {
    NSLog(@"The cat sounds");
  }
  -(void) sleep {
    NSLog(@"The cat is sleeping");
  }
@end

// main.m
#import <Foundation/Foundation.h>
#import "Cat.h"

int main(int argc, const char* argv[]) {
    @autoreleasepool {
        Cat* myCat = [[Cat alloc] init];
        myCat.name = @"Tony";
        [myCat makeSound];
    }
```

```
    return 0;
}
```

Analice el código dado y determine qué características tiene el sistema de tipos de dicho lenguaje en base a lo observado. Sí algún elemento no puede determinarse con la información provista, investigue y conteste. Analice en términos de los ejes vistos en la teoría.

- Momento de la verificación
- Declaratividad de los tipos
- Identificación de los tipos

Ejercicio 2

El código siguiente corresponde al lenguaje de programación Ruby.

```ruby
class Element
    def symbol
        puts "this method returns the symbol of the element"
    end
end

class Sodium < Element
    def symbol
        puts "symbol for Sodium: Na"
    end
end

class Copper < Element
    def symbol
        puts "symbol for Copper: Cu"
    end
end
```

Analice el código dado y determine qué características tiene el sistema de tipos de dicho lenguaje en base a lo observado. Sí algún elemento no puede determinarse con la información provista, investigue y conteste. Analice en términos de los ejes vistos en la teoría.

- Momento de la verificación
- Declaratividad de los tipos
- Identificación de los tipos

Ejercicio 3

El código siguiente corresponde al lenguaje de programación Scala.

```
object App {

  case class Post(title: String,
                  body: String,
                  tags: List[String])

  case class Address(street: String,
                     city: String,
                     state: String,
                     zip: String,
                     tags: List[String])

  def formattedTags(taggable: { def tags: List[String] }) = {
    taggable.tags.mkString(",")
  }

  def main(args: Array[String]) {
    val post = Post("Weekly NFL Scores",
                    "This week...",
                    List("news", "sports", "local"))
    println(formattedTags(post))    // news,sports,local

    val address = Address("1 Main St.",
                          "Anytown",
                          "CA",
                          "12345",
                          List("home", "work"))
    println(formattedTags(address)) // home,work
  }
}
```

Analice el código dado y determine qué características tiene el sistema de tipos de dicho lenguaje en base a lo observado. Sí algún elemento no puede determinarse con la información provista, investigue y conteste. Analice en términos de los ejes vistos en la teoría.

- Momento de la verificación
- Declaratividad de los tipos
- Identificación de los tipos

Ejercicio 4

Considere los tipos “A”, “B”, “C” y “D”, y que existen los valores “a”, “b”, “c” y “d”, donde “a” es de tipo “A”, “b” es de tipo “B”, “c” de tipo “C” y “d” de tipo “D”. Además, tenga en consideración las siguientes relaciones de subtipado:

- B es subtipo de A
- C es subtipo de A

- D es subtipo de C

Luego, indique cuáles de estas asignaciones son válidas y cuáles no.

a. var x : A := a
b. var x : B := a
c. var x : C := a
d. var x : D := a
e. var x : A := b
f. var x : B := b
g. var x : C := b
h. var x : D := b
i. var x : A := c
j. var x : B := c
k. var x : C := c
l. var x : D := c
m. var x : A := d
n. var x : B := d
o. var x : C := d
p. var x : D := d

Ejercicio 5

El siguiente es código Scala. En el mismo se expresan tipos paramétricos con varianza de distinta índole.

```
class Thing
class Vehicle extends Thing
class Car extends Vehicle
class Jeep extends Car
class Coupe extends Car
class Motorcycle extends Vehicle
class Bicycle extends Vehicle
class Tricycle extends Bicycle
class Parking[A >: Bicycle <: Vehicle](val plaza: A)
```

Se pide que determine sí las siguientes son instancias válidas del tipo o no lo son.

1. Parking[Thing]
2. Parking[Car]
3. Parking[Vehicle]
4. Parking[Bicycle]
5. Parking[Tricycle]