Kotlin exercises

Note: See the skeleton files for the exercises

1. Functions in Kotlin - The createCounter() function declare a pair of functions for managing a counter. Instead of creating a class, we manipulate the counter using only functions.
In createCounter() define a local property counter. Then define two local functions:

- inc(): increases the counter value by one
- value(): returns the value of counter

Finally, return a pair of function references to these local functions:

return Pair(::inc, ::value)

Pair is a class in Kotlin library.
The two local functions capture the same local variable counter. This style of managing state is commonly used in pure functional programming languages that don't have classes.

Test your functions by calling inc() 10 times, and see if value() return the correct answer.

2. Complete the implementation of the createContainer() function. It returns a Pair of functions to control a single value Int container. The first function puts an Int element into the container, the second function removes the element from the container and returns it. If the container is empty, the second function returns null.

Test it calling add() and remove() in several ways.

3. The starter code includes a data class called Pet containing a var String property name and an enum property habitat. The Habitat enumeration can be LAND, WATER or AMPHIBIOUS, and it also contains a member function livesIn(pet: Pet) that tests to see whether pet lives in a particular Habitat.

The main() starter code creates a List<Pet>. Using filter() together with member references, implement three functions liveOnLand(), liveInWater() and areAmphibious() that discover which pets in the list live on land, in the water, or are amphibious. Lastly, use partition() to implement the partitionAmphibious() function that divides the pets into those that are amphibious and those that are not.

filter() and partition() are on the library for Collection classes

4. To illustrate different types of member references, consider three characteristics of natural numbers: whether a number is even, whether it's prime (doesn't have divisors other than 1 and itself), and whether it's perfect. A perfect number means that the sum of all the divisors

(excluding the number itself) equals the number. For example, 6 is a perfect number: 6 = 1 + 2 + 3, where 1, 2, 3 are the divisors.

isEven is an extension property, isPrime() is an extension function and isPerfect() is a top-level function. Your task is to complete the implementations and pass the corresponding property or function reference to different filter invocations in main().

5. Classes - Create a class Floating that contains a property d of type Double. Initialize d in the constructor. Include a toString() member function that returns the contents of d. Write a main() to exercise the Floating class.

6. Build a Robot class that should walk in a square field with coordinates 0 to fieldSize-1 (steps). Instances should have the position x and y, and the fieldSize. The class should have functions goLeft(), goRight(), goUp(), and goDown() to displace the Robot k steps in each direction. It should never walk outside of the field, but suppose it walk-around (when it reaches one side it reappears on the opposite side).

Also include a report function (getLocation()), returning a String: x, y

When an instance of Robot is printed it should print: Robot(x=…, y=…)

Test thoroughly the class in the main() function