Functional Programming: Lists

This lesson introduces the List structure, which is one of the most widely used structures in functional programming.

1. Without explicit type annotations, using the operators `::` and `:::`, any member of the class `List` and considering only the values defined by the following statements:

```
val aList:List[Int] = List(1, 2, 3)
val bList=List("edom", "odsoft", "tap")
val cList=List('a', 'b')
val dList=List(true, false)
val e=5.6
val fList = List(1.0, 2, 3)
val g='i'
```

   provide results of the indicated type:

   a. `List[Char]` using `bList`
   b. `List[Int]` using `bList`
   c. `List[Any]` using only `aList` and `bList`
   d. `List[Any]` using `e`
   e. `List[AnyVal]` using `e`

   Use Scala documentation, namely http://docs.scala-lang.org/tutorials/tour/unified-types.html. No other values besides the ones in the statements can be used.

2. What are the expressions to specify in order to obtain `List(1, 3, 5, 7, 9)` using

   a. `List.range`
   b. `List.tabulate`

3. Create the expressions to obtain `List("Joana", "José")` from `List("Maria", "Ana", "Joana", "Julia", "Paulo", "José")` using

   a. `filter`
   b. `for… yield`

4. What should be the value of x to allow the indicated result?

```
val x = ???
List(1, 2, 3, -1, -2, -3, 0).map(x)
//> Result: List[Int] = List(1, 2, 3, 1, 2, 3, 0)
```

5. Define the function `hasDigit` that verifies if what is passed has any digit, without cycles, var or val, declarations or recursion. It must work with the test examples provided.

6. Consider a Scala function `max_` to be completed:

```
def max_(xs: List[Int]): Option[Int] = {
  @tailrec
  def maxAux(m: Int, ys: List[Int]): Option[Int]  = ys match {
    case Nil => ???
    case x :: t => ???
  }
  ???
}
```

   Replace the expression `???` to obtains the max element of a `List[Int]`.

7. Consider these calls:

```
applyF_(List(19, 2, 3),List("aa", "b", "c")) (f1)
applyF_(List(19, 2, 3),List("aa", "b", "c")) (f2)
applyF_(List(19, 2, 3),List("aa", "b", "c")) (f3)
applyF_(List(19, 2, 3),List("aa", "b", "c")) (f4)
```

   with, respectively, the following results:

```
res41: String = 19aa2b3c
res42: String = 1923aabc
res43: String = 3219aabc
res44: String = 3291aabc
```

   Specify the functions, knowing that the first argument list of `applyF_`should specify two lists of generic types and its result should be a String. The arguments of each of the other functions (`f1,f2,f3,f4`) should be two lists of generic types and their result should be of type String.