Algorithms and Computational Thinking
Autumn 2017

Tuesday, 19th December 2017

Understanding Functional Programming

Objective : In the two following exercises, you will get a basic understanding of the functional programming.

Exercise 1

If we go back to the last slide of the lecture, there is an example of the use of functions as parameters in the context of the creation of matrices. The goal of this first exercise is to implement the creation of several dense matrices with different functions passed as parameters. You will also print the result of all created dense matrices as described in the last slide of the lecture. You must do this exercise in Scala and Swift. Instead of creating a complete hierarchy of classes of matrix, we will simply use the implementation of the DenseMatrix class depicted below (follow the same structure for your implementation in Swift).

```
class DenseMatrix(private val rows: Int, private val cols: Int) {

  private val matrix = Array.ofDim[Int](rows, cols)

  def this (rows: Int, cols: Int, init: (Int,Int) => Int) {
    this(rows,cols)
    for (i <- 0 to rows - 1; j <- 0 to cols - 1)
      this.matrix(i)(j) = init(i,j)
  }

  def print(): Unit = {
    println("DenseMatrix")
    for (i <- 0 to rows - 1) {
      var strMatrixLine = "| "
      for (j <- 0 to cols - 1)
        strMatrixLine = strMatrixLine + matrix(i)(j) + " "
      strMatrixLine = strMatrixLine + "|"
      println(strMatrixLine)
    }
  }

}
```

Exercise 2

You will discover how to pass a function to another function in Scala and Swift in order to add several numbers. These numbers can be integers, square numbers or power of two numbers that we can obtain by using the three functions given in Scala below :

```
def id(x : Int) : Int = x
def square(x : Int) : Int = x * x
def powerOfTwo(x : Int) : Int = if (x == 0) 1 else 2 * powerOfTwo(x - 1)
```

To reach the main goal of the second exercise, we need to implement a function *sum* that has the signature detailed below. This function takes a function as well as two integers as parameters. The function f must be one of the three functions above to generate numbers to sum and the two integers are the boundaries of the sum, i.e., we start with a and end with b.

```
def sum(f : Int => Int, a : Int, b : Int) : Int = { /*your implementation*/ }
```

In order to test your code and display the final results, you can follow the lines of code below given in Scala.

```
println("sum int numbers 1 to 6 = " + sum(/*your parameters*/))
println("sum square numbers 1 to 6 = " + sum(/*your parameters*/))
println("sum power of two numbers 1 to 6 = " + sum(/*your parameters*/))
```

You must also do this second exercise in Scala and Swift.