Tutorial
Functional Programming

Question 1.

Write a function **lstSquare(n:Int)** that returns a list of the squares of the numbers from 1 to n?
For example:
lstSquare(5) returns List(1,4,9,16,25)

a) Use recursive approach?

b) Use high-order function approach?

Question 2.

Write a Scala function **pow**(x: Double, n: Int): Double that return a power x^n. (don't use Scala library math).
For example:
pow(2,0) returns 1
pow(2,3) returns 8

a. Use recursive approach

b. Use high-order function approach

Question 3.

Make a list of the elements 1, 2, 3, 5, 7.
Write your own functions (don't use Scala library functions: append, reverse):
append(a: List[Int], b: List[Int]): List[Int]
reverse(a: List[Int]): List[Int]

a. Use recursive approach

b. Use high-order function approach

Question 4.

Write a Scala function **lessThan**(n: Int, lst: List[Int]) that returns a list of all numbers in lst less than n. For example, lessThan(50, List(1, 55, 6, 2)) yields List(1,6,2)

a. Use recursive approach

b. Use high-order function approach

Question 5.

Assume that there are many different data structures which contain a component in String type. Write your own function **lookup** that can find in a list an element whose component is the specified string.

Hint: The prototype of the function should be:

lookup[T](str:String,lst:List[T],ret: T => String):Option[T]

For example, assume that there are the following classes:

case class A(n:String,v:Int)
case class B(x:Int,y:A)
lookup("m",List(A("n",3),A("m",5)),(x:A)=>x.n) returns Some(A("m",5))