CY TECH sciences et techniques	**Postgraduate level - Second year** **Exercise sheet #2 – Functions**	
	Subject : **Functional programming**	*Date* : **2020 – 2021**
		Duration : **3 heures**
		Number of pages : **2**

Exercise 1 (Gestion du calendrier)**.**

a. Define a function `isLeap(year : Int) : Option[Boolean]` that :

- checks if `year` is a leap year ;
- returns `None` if `year` is negative.

b. Define a function `numberOfDays(leap : Boolean, month : Int) : Option[Int]` that :

- computes the total number of days of `month`$^{\text{th}}$ month of the year ; `leap` whether considered year is a leap year or not ;
- returns `None` if `month` is invalid.

c. Explain why the signature of `numberOfDays` may be improved.

Exercise 2 (Computation of π)**.**

a. It is known that $\lim_{n\to+\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$. Use this fact to compute an guess of π in a function `pi` with parameter n. *No loop is allowed. Any recursive computation must be tail-recursive.*

[*Hint* : `math.sqrt` is the square root function.]

b. We consider a Monte-Carlo method :

- pick n random points in the square $[0,1]^2$;
- count the number of points inside the $\{(x,y)\in[0,1]^2,\ x^2+y^2\leq 1\}$ (quarter of disk with unit radius)

It is known that the probability that a point of $[0,1]^2$ lies in the set $\{(x,y)\in[0,1]^2,\ x^2+y^2\leq 1\}$ is $\frac{\pi}{4}$.
Use this fact to compute a guess of π with above method in a function `piMC` with parameter n. *No loop is allowed. Any recursive computation must be tail-recursive.*

Generating a random double between 0 and 1 is similar to Java :

(a) first, create a random generator : `val rand = new util.Random()` ;

(b) then, when needed, compute the next value with `rand.nextDouble()`.

c. Is `pi` a pure function ? Is `piMC` a pure function ?

Exercise 3. (From online course *Principles of functional programming using Scala* by Martin ODERSKY)

a. Let I be an interval of $\mathbb{R}$ and f a real function differentiable on I.

One can define the NEWTON method as the recurring sequence $(x_n)_{n\in\mathbb{N}}$ defined by following relation :

$$\forall n \in \mathbb{N},\ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

It is assumed that $(x_n)_{n\in\mathbb{N}}$ converge a root of f.

Write the tail-recursive function

```
squareRoot(a : Double, eps : Double) : Double
```

that computes a guess of the square root of `a` with the NEWTON method applied to $f : x \mapsto x^2 - \mathtt{a}$ with $x_0 = 1$. `eps` is the precision allowed on the square of the approximation.

b. We aim at generalizing the iterative process. Write the tail-recursive function

```
guess[T](isGoodEnough : T => Bool, improveGuess : T => T)(initialGuess : T) : T
```

that computes a guess :

- `isGoodEnough` is the termination test, returning `true` if guess is… good enough (!) ;
- `improveGuess` is the function used to improve guess at each step ;
- `initialGuess` is… the initial guess (!).

(i) Redefine `squareRoot` using `guess`.

(ii) Use `guess` to compute a guess of π as a root of sine function using NEWTON method.

c. Let I be an interval of $\mathbb{R}$ and ϕ an application defined on I such that $\phi(I) \subset I$.

One can defined the fixed-point method as the recurring sequence $(x_n)_{n\in\mathbb{N}}$ defined by the following relation :

$$x_0 \in I \quad \text{et} \quad \forall n \in \mathbb{N},\ x_{n+1} = \phi(x_n)$$

It is assumed that $(x_n)_{n\in\mathbb{N}}$ converge to a fixed point of ϕ, i.e. an element x of I such that $\phi(x) = x$.

(i) Use `guess` to write the function

```
fixedPoint(gap : Double, f : Double => Double)(initialGuess : Double) : Double
```

that computes a guess of a fixed point of `f` starting with `initialGuess` as initial guess. `gap` is the precision allowed between consecutive guesses.

(ii) Write the function

```
damping : (f : Double => Double)(x : Double)
```

that computes the damped function of `f` at `x`, i.e. $\dfrac{\mathtt{x} + \mathtt{f(x)}}{2}$.

Verify mathematically that any fixed point of a function is a fixed point of its damped function and conversely.

(iii) Use `fixedPoint` and `damping` to write the function

```
fixedPoint_damping(gap : Double, f : Double => Double)(initialGuess : Double) : Double
```

that computes a guess of a fixed point of `f` by damping starting with `initialGuess` as initial guess. `gap` is the precision allowed between consecutive guesses.

(iv) Identify `squareRoot` as a particular case of `fixedPoint_damping`.