CY TECH sciences et techniques	**Postgraduate level - Second year** **Exercise sheet #4 – Collections**	
	Subject : **Functional programming**	*Date* : **2025**
		Duration : **3 heures**
		Number of pages : **3**

Exercise 1.

a. Rewrite method `Seq.++` using one of `foldLeft`, `foldRight` or `fold`.

b. Define a method `translate0(dict : Map[String, String], french : String) : String` that performs a low-level translation with :

- `dict` as the french to english dictionary ;
- `french` as the french sentence with no punctation.

(*Hint*: You may use `String.split` and `Iterable.mkString`.)

```
scala> val dict = Map("le" -> "the", "chat" -> "cat", "mange" -> "eats",
                      "souris" -> "mouse", "la" -> "the")
dict : Map[String, String] = ...

scala> translate0(dict, "Le chat mange la souris")
res0: String = "the cat eats the mouse"
```

c. Define a method `occurrences(s : String) : Map[String, Int]` that computes the number of each word's occurrences in the sentence `s`.

Exercise 2. The IT department of CY Tech wants to manage the users of her resources :

- for each user, is stored his last and name and age ;
- for any student, is also stored his school year ;
- for any teacher, is also stored his teaching department and his seniority.

a. Create a new sbt project.

(i) Create package `model` in subdirectory `src/main/scala/model` and define the model.
(*Hint*: Use case classes as much as possible.)

.

(ii) Given the collection of students (named `students`) and teachers (named `teachers`), write in Scala the following requests :

- search for all students of a given school year and sort them ;
- search for all teachers of a given teaching department and sort them ;
- search for all teachers with at least ten years of seniority and sort them ;
- search for all couples student/teacher where student has the same name as teacher ;
- compute for any school year the number of students ;
- compute for any teaching department the number of teachers.
- compute for any school year the mean age of students ;
- compute for any teaching department the mean age of teachers ;
- compute for any value of seniority the mean age of teachers.

These requests should never throw an exception.

b. The education department is interested in using this model adding also courses. For any course, is stored :

- the teacher doing the course ;
- the school year attending the course ;
- the room where the course takes place : a room is identified by its name, the building where it is located and its capacity ;
- the starting time of the course ;
- the duration of the course (in minutes).

(i) Update the model.

(ii) Given the collections of courses (named `courses`), write in Scala the following requests :

- search for all courses of a given teacher and sort them by start time ;
- search for all courses of a given student and sort them by start time ;
- compute for any room the number of courses ;
- compute for any building the number of courses ;
- search for any course in a room that does not fit with headcount ;
- search for any conflit in `courses` (courses are planned on fixed non-overlapping times).

These requests should never throw an exception.

Exercise 3 (For the bold ! ™)**.**

a. *(i)* Write in Scala the method

```
recurrence[T](f : (Int, T) => T)(initial : (Int, T)) : LazyList[(Int, T)]
```

that builds the recurrent sequence $(n, u_n)_{n \geq n_0}$ defined by (n_0, u_{n_0}) = `initial` and u_{n+1} = `f`(n, u_n) for any natural integer n.

(ii) Rewrite `LazyList.from` using `recurrence`.

b. *(i)* Write in Scala the method

```
discrete(law : Map[Int, Double]) : Option[Int]
```

simulating a natural integer-valued random variable whose probability distribution is given by `law`. Should `law` be empty, `None` is returned. (*Hint* : You may use `Iterable.scanLeft`.)

(ii) Modify `discrete` so that `law` has type `List[(Int, Double)]` without using conversions.

(iii) Modify `discrete` so that `law` has type `LazyList[(Int, Double)]`.

c. *(i)* Use code in question b.*(iii)* to simulate a geometric law and a POISSON law.

(ii) What about using codes in questions b.*(i)* and b.*(iii)* for these laws ?