CY TECH sciences et techniques	**Postgraduate level - Second year** **Exercise sheet #3 – Classes**	
	Subject : **Functional programming**	*Date* : **2025**
		Duration : **3 heures**
		Number of pages : **2**

Exercise 1 (Binary trees)**.**

a. A generic binary tree is either :

- an empty tree ;
- or a node with a label and left and right childs.

Type must provide following methods :

- `size` computes number of labels of tree ;
- `height` computes tree height ;
- `contains(x : T)` checks if `x` is a label in tree ;

b. Try to write `size`, `height` and `contains` as three special cases of one new method named `fold`.

Exercise 2 (Rational numbers)**.**

a. Define the `Rational` type representing rational numbers with :

- exactly one constructor that may be used without keyword `new` :
  - `Rational(numerator : Int, denominator : Int)` ;
- arithmetic unary operators `unary_-`, `inverse` ;
- arithmetic binary operators `+`, `-`, `*`, `/` ;
- equality operator ;
- comparison operators using predefined `Ordered` trait ;
- `toString` redefined accordingly ;

b. Modify type `Rational` such that arithmetic binary operators, equality and comparison operators can handle mixed parameters, i.e. with type `Int` and `Rational`.

Exercise 3 (Arithmetical expressions)**.** An `expression` type describes expression trees with :

- integer and rational constants ;
- usual arithmetic operators (+, −, ∗ and unary −) ;

Type must be designed such that *any built instance represent a valid expression*.

Type must provide following methods :

- `eval` computes the value of the expression ;
- `+`, `-`, `*` and `unary_-` builds the corresponding expression.

Hint : You may start with integer constants only at first.