# Exercice 8 :

1. La différence entre les fonctions et les méthodes dans le contexte de la programmation fonctionnelle en scala.

Scala a des fonctions et des méthodes, à la fois sur la petite différence dans la sémantique. Méthode Scala fait partie d'une classe, et la fonction est un objet qui peut être affecté à une variable. En d'autres termes, la fonction définie dans une classe qui est une méthode.

2. Considérons un cas d'utilisation de la récursivité et essayons de l'appliquer

```
scala> def reverse(l : List[Int]) : List[Int] = {
     | l match{
     | case List() => List()
     | case x::xs => reverse(xs):::List(x)
     | }}
reverse: (l: List[Int])List[Int]
```

La fonction reverse cherche deux motifs dans la liste qui lui est passée : soit la liste vide, auquel cas elle n'a rien à faire et renvoie une liste vide, soit la liste contient au moins un élément. Dans ce deuxième cas, la fonction enlève la tête de la liste et s'appelle récursivement en passant en paramètre la liste privée de son premier élément. Lorsque l'appel récursif est terminé, la fonction concatène la liste renvoyée à une liste contenant seulement la tête précédemment enlevée et retourne le résultat.

3. Essayons d'utiliser des fonctions dans des fonctions.

```
scala> def convertToUpper(name:String):String = name.
     | toUpperCase
convertToUpper: (name: String)String

scala> def convertToLower(name:String):String = name.
     | toLowerCase
convertToLower: (name: String)String

scala> def changeCase(givenName:String,caseConverter:(String)=>
     | String) = caseConverter(givenName)
changeCase: (givenName: String, caseConverter: String => String)String

scala> changeCase("bigdata",convertToUpper)
res17: String = BIGDATA

scala>
```

4. Comprenons si les variables ont été copiées par valeur ou par référence dans scala

Scala utilisent exclusivement l'appel par valeur. Incidemment, l'appel par nom de Scala est implémenté en utilisant l'appel par valeur, la valeur étant un (pointeur vers un) objet fonction qui renvoie le résultat de l'expression.

5. Comprenons les meilleures pratiques des fonctions, en ce sens qu'elles doivent être conçues de manière à effectuer une et une seule tâche.

En Scala, une fonction est une « valeur de première classe ». Comme n'importe quelle autre valeur, elle peut être passée en paramètre ou renvoyée comme résultat. Les fonctions qui prennent en paramètres d'autres fonctions ou qui les renvoient comme résultat sont appelées fonction *d'ordre supérieur*.

- Écrivons une fonction additionnant tous les entiers compris entre deux bornes a et b :

```
scala> def sumInts(a: Int, b: Int): Int =
     |  if (a > b) 0 else a + sumInts(a + 1, b)
sumInts: (a: Int, b: Int)Int

scala> sumInts(1,3)
res18: Int = 6

scala>
```

- Ecrivons une fonction additionnant les carrés de tous les entiers compris entre deux bornes a et b :

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

def square(x: Int): Int = x * x
def sumSquares(a: Int, b: Int): Int =
    if (a > b) 0 else square(a) + sumSquares(a + 1, b)

// Exiting paste mode, now interpreting.

square: (x: Int)Int
sumSquares: (a: Int, b: Int)Int

scala> sumSquares(1,3)
res0: Int = 14

scala>
```