Exercice 9 :

EXERCICES : LES LISTES ET LA FONCTION CARTE

1. Créer une liste de numéros et retourner vrai si un élément est pair ; sinon, retourner faux.

- Si paire

```
scala> val listOfNumbers = List(2,4,6,8)
listOfNumbers: List[Int] = List(2, 4, 6, 8)

scala> for (i<-listOfNumbers){
     | if(i%2==0){
     | println("true")
     |  }
     | else{
     | println("false")
     | }
     | }
true
true
true
true
```

- Si impaire

```
scala> val listOfNumbers = List(1,3,5,7)
listOfNumbers: List[Int] = List(1, 3, 5, 7)

scala> for (i<-listOfNumbers){
     | if(i%2==0){
     | println("true")
     | }
     | else {
     | println("false")
     | }
     | }
false
false
false
false

scala>
```

2. Créons une liste de chaînes et extrayons le premier et le dernier caractère de chaque chaîne.

```
scala> val myStringList = "bigdata"
myStringList: String = bigdata

scala> myStringList(0)
res1: Char = b

scala> myStringList(6)
res2: Char = a

scala>
```

3. Chargeons un fichier en scala et chargeons son contenu dans une liste

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

import java.io._
object Test {
   def main(args: Array[String]) {
      val writer = new PrintWriter(new File("test.txt" ))

      writer.write("j'apprend du bigdata avec scala")
      writer.close()
}
}

// Exiting paste mode, now interpreting.

import java.io._
defined object Test
```

EXERCICES : LISTES

1. Explorons comment ajouter deux listes, ajoutons un élément à une liste et faisons une union ou une intersection de deux listes.

```
scala> val list1:List[String]=List("test1","test2")
list1: List[String] = List(test1, test2)

scala> val list2:Option[List[String]]=Some(List("test1"))
list2: Option[List[String]] = Some(List(test1))

scala> list2.getOrElse(List.empty).intersect(list1)
res0: List[String] = List(test1)

scala>
```

2. Créons une liste de chaînes, puis essayons d'utiliser reduce() pour obtenir le même résultat que vous obtiendriez en utilisant le .mkString().

```
scala> val myStringList = List("scala","java","spark","bigdata")
myStringList: List[String] = List(scala, java, spark, bigdata)

scala>  myStringList.mkString(";")
res4: String = scala;java;spark;bigdata

scala> myStringList.reduce((x,y)=>x+";"+y)
res5: String = scala;java;spark;bigdata

scala>
```

EXERCICE : LISTBUFFEERS

1. Essayons d'utiliser ArrayBuffer comme une collection modifiable.

ArrayBuffer est recommandée comme classe à usage général pour les collections séquentielles *modifiables.* (ArrayBuffer est une collection séquentielle indexée)

- Créons un ArrayBuffer avec des éléments initiaux :

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

import scala.collection.mutable.ArrayBuffer
val nums = ArrayBuffer(1, 2, 3)

// Exiting paste mode, now interpreting.

import scala.collection.mutable.ArrayBuffer
nums: scala.collection.mutable.ArrayBuffer[Int] = ArrayBuffer(1, 2, 3)

scala>
```

- Comme les autres classes de collection mutables, nous pouvons ajouter des éléments à l'aide des méthodes +=et : ++=

Ajoute un élément

```
scala> nums += 4
res6: nums.type = ArrayBuffer(1, 2, 3, 4)
```

Ajoute deux éléments ou plus (la méthode a un paramètre var args)

```
scala> nums += (5, 6)
res7: nums.type = ArrayBuffer(1, 2, 3, 4, 5, 6)
```

Ajoute des éléments d'une autre collection

```
scala> nums ++= List(7, 8)
res8: nums.type = ArrayBuffer(1, 2, 3, 4, 5, 6, 7, 8)
```

EXERCICES SUPPLÉMENTAIRES

1. Recherchons Array en scala. En quoi est-elle différente des autres collections que nous avons étudiées :

Array est un type spécial de collection en scala. C’est une structure de données de taille fixe qui stocke des éléments du même type de données. L'indice du premier élément d'un tableau est zéro et le dernier élément est le nombre total d'éléments moins un. C'est une collection de valeurs mutables.

2. Vector en scala et en quoi est-il différent des autres collections que vous avez étudiées.

Les vecteurs permettent d'accéder à n'importe quel élément de la liste en temps "effectivement" constant. C'est une constante plus grande que pour accéder à la tête d'une liste ou pour lire un élément d'un tableau, mais c'est quand même une constante. C’est un nouveau type de collection dans Scala 2.8 qui résout l'inefficacité de l'accès aléatoire aux listes.