Durée du CC : 1h15

Prénom ..... Nom : .....

Les réponses sont à apporter directement sur la feuille, en dessous de la question correspondante. Le bénéfice du doute ne sera pas accordé en cas de réponse illisible.

Si un appareil électronique est manipulé durant le CC, un dossier sera envoyé au conseil disciplinaire, peu importe la justification donnée. Toute sortie est définitive. La fiche-triche est le seul document autorisé.

Les sections sont indépendantes.

1 Listes

Question 1 Implémentez une fonction

```
produit(liste: List[Int]) : Int
```

qui renvoie le produit des éléments de `liste`. Vous devrez utiliser un pattern matching au cœur de cette fonction.
Ainsi, `produit([2,3,2])` renverra l'entier `12`. On rappelle que par une convention bien pratique, le produit de zéro entiers entre eux vaut `1`.

Question 2 Implémentez une fonction

```
mymap[T,U](f : T=>U, liste: List[T]) : List[U]
```

qui renvoie la liste obtenue en appliquant la fonction `f` à chaque élément de la liste `liste`. Vous devrez utiliser un pattern matching au cœur de cette fonction (en particulier, l'utilisation de `map` est interdite).
Sur les arguments `(x:Int)=>2*x` et `[1,2,3]`, `mymap` renverra donc la liste `[2,4,6]`.

2 HashMap

Question 3 Écrivez une fonction `double[K]` qui prend en arguments un `HashMap[K,Int] h` (dont les valeurs sont donc des entiers), et qui retourne un nouveau `HashMap[K,Int]` contenant les mêmes clefs que `h`, et où la valeur associée à la clef `k` est le double de la valeur associée `k` dans `h`.
Par exemple :

```
val h = new HashMap[String,Int]( "a"->2 , "b"->3 )
val hd = double[String](h)
# hd contient les clefs/valeurs "a"->4, "b"->6
```

3 Tamis

Question 4 Créez une classe `Tamis[T]` paramétrée par le type `T`, ayant les caractéristiques suivantes :

- son constructeur principal attend une fonction `p:T=>Boolean`, qui sera un attribut non-mutable des objets de type `Tamis`,
- elle possède un constructeur secondaire n'ayant pas d'argument, qui considère `p` comme étant la fonction qui a tout `x` associe `true`,
- étant donné un objet `tamis` de type `Tamis[T]` et une liste `l` contenant des éléments de type `T`, faites en sorte que `tamis(l)` renvoie la liste obtenue à partir de `l` en ne conservant (sans changer leur ordre) que les éléments `x` pour lesquels `tamis.p(x)` vaut `true`.

4 Covariance

On rappelle qu'en Scala, toutes les classes héritent de la classe `Any` – c'est l'équivalent de la classe `Object` en Java. Considérons le code Scala suivant :

```scala
val lint : List[Int] = 10::Nil
val lany : List[Any] = lint
lany[0] = false
val o : Option[Int] = lint match {
  case Nil => None
  case n::_ => Some(n)
}
```

Question 5 Pourquoi, s'il était correct, ce code serait-il problématique ?

Question 6 À quelle ligne se trouve l'erreur, et pourquoi n'est-ce pas possible en Scala ?

Question 7 Java a pris un parti différent pour ses `ArrayList<T>`. Quel est l'avantage de la manière de faire de Scala ?