Examen du 08/02/2014 (1h)

TOP : Techniques and tOols for Programming – 1A

Documents interdits, à l'exception d'une feuille A4.

La notation tiendra compte de la présentation et de la clarté de la rédaction.
Le barême, approximatif, est donné sur 10 points puisque l'épreuve dure 1h ($\frac{10}{20} = 0,5 = \frac{1h}{2h}$).

★ Exercice 1: Complexité algorithmique (3pt).

▷ **Question 1:** (1pt) Étudiez le nombre d'additions réalisées par les algorithmes suivants dans le meilleur cas, le pire cas, puis dans le cas moyen en supposant que les tests ont une probabilité de $\frac{1}{2}$ d'être vrai.

Listing 1.a

```
1 for (i <- 1 to n) {
2    if (T(i)>a) {
3        s += T(i)
4    }
5 }
```

Listing 1.b

```
1 if (a > b) {
2   for (i <- 1 to n) {
3     x += a
4   }
5 } else {
6    x += b
7 }
```

▷ **Question 2:** (2pt) Donnez la complexité des programmes suivants. Vous donnerez une borne supérieure avec un $O()$ dans un premier temps, puis vous affinerez votre calcul en utilisant la notation $\Theta()$.

Listing 2.a

```
1 for (i <-  5 to n-5) {
2    for (j <- i-5 to i+5) {
3      x += 3
4    }
5 }
```

Listing 2.b

```
1 for (i <- 1 to n) {
2     for (j <- 1 to i) {
3           for (k <- 1 to j) {
4                 x += 4
5           }
6     }
7 }
```

★ Exercice 2: La dichotomie (4pt).

▷ **Question 1:** Écrivez une fonction **récursive en scala** cherchant l'indice d'un élément donné dans un tableau trié. La recherche doit être dichotomique. Le prototype de la fonction doit être le suivant :

```
def dicho(tab:Array[Int], elm:Int):Int
```

La fonction doit retourner l'indice où se trouve l'élément dans tab s'il y est, ou -1 sinon.

▷ **Question 2:** Calculez la complexité de cette fonction.

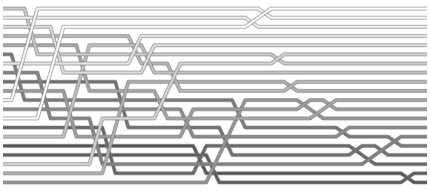
▷ **Question 3:** Montrez la terminaison de cette fonction.

▷ **Question 4:** Dérécursivez la fonction précédente, en justifiant ce que vous faites et pourquoi vous avez le droit de le faire. Le programme résultant doit être écrit en scala.

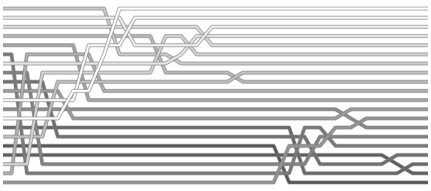
★ Exercice 3: Identification d'algorithmes de tri (d'après Aldo Cortesi – 3pt).

Les schémas suivants montrent le fonctionnement de divers algorithmes de tris. Chaque trait grisé indique une valeur, et l'axe des abscisses montre le temps qui passe tandis que l'axe des ordonnées montre la position de chaque valeur (=trait) dans les différentes cases du tableau. La case n°1 est en haut, et la case n°20 est en bas, et une couleur plus claire signifie une valeur plus petite. Quand deux traits se croisent, c'est que l'algorithme a inversé les deux valeurs à cet instant.

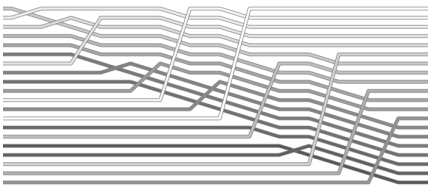
▷ **Question 1:** Identifiez le comportement des algorithmes suivants : tri à bulle, tri par insertion, tri par sélection, shell sort, quick sort. Argumentez vos réponses.



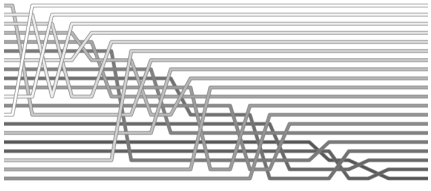
Algorithme A.



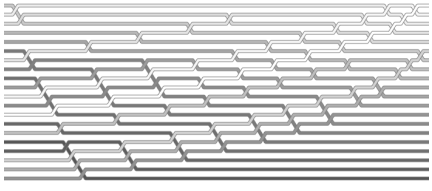
Algorithme B.



Algorithme C.



Algorithme D.



Algorithme E.