Université Denis Diderot **Types de données et objets**
L1 S2 **2010 – 2011**

Examen du 18-Mai 2011. Durée 3 heures.
Le barême est indicatif et est susceptible d'être réajusté.
Il sera tenu compte de la qualité et de la clarté de votre copie (± 2 points).

A/- Listes chaînées (2 points).

Dans cet exercice, on considère des listes chaînées.

1) Donner le type abstrait d'une liste chaînée.
2) Ecrire une classe liste chaînée en Java.
3) Ecrire une méthode permettant de fusionner deux listes triées d'entiers en une seule liste triée.
4) En déduire une méthode permettant de trier une liste d'entiers.

B/- Arbres binaires complets, récursion et complexité (6 points).

Un arbre binaire est dit *complet* si chacun de ses nœuds (sommets) possède soit 0 fils soit 2 fils.

1) Dessiner les **neuf** arbres binaires complets avec 1, 3, 5, et 7 sommets. Montrer qu'il n'existe pas d'arbres binaires complets avec $2n, n > 0$ sommets.
2) Expliquer comment obtenir *tous* les arbres binaires complets avec $2n + 1$ sommets.
3) En déduire un algorithme (en pseudo-code) récursif **touslesArbres** construisant tous les arbres binaires complets avec n sommets (n quelconque). Expliciter les *conditions d'arrêt* et le *corps de la récursion* de votre fonction.
4) Soit T_n la complexité de la fonction **touslesArbres** de la question précédente. Sans la résoudre, donner une équation récursive satisfaite par T_n.

C/- Tableau de piles, tas et tri (4 points).

On dispose d'un tableau T de p piles d'entiers (les éléments de T sont donc des piles dont chacune contient des entiers). On suppose que T vérifie les propriétés suivantes.

(P_1) Aucune pile n'est vide et il y a n entiers réparties dans les p piles.
(P_2) Les sommets de piles dans T vérifient

$$\text{Sommet}(T[0]) \geq \text{Sommet}(T[1]) \geq \cdots \geq \text{Sommet}(T[p-1]).$$

(P_3) Dans chaque pile, les entiers sont ordonnés du plus grand au plus petit (le plus grand est donc au sommet de la pile).

1/ Montrer que la propriété (P_1) implique $p \leq n$.
2/ Décrire un algorithme (en pseudo-code) qui transforme un tel tableau T ayant p piles et n entiers en un tableau trié contenant les n entiers des p piles.

3/ Ecrire une classe **PileEntier** et les méthodes qu'elle devrait contenir.

4/ Ecrire alors l'algorithme de la question C/2/ en Java.

5/ Conclure en comparant avec les méthodes de tri connues.

D/- Des sommes (3 points).

On dispose du code Java suivant

```
public class Ducode {
   public static void sommePre(int  []t, int n) {
      for (int j = n-1; j>=0; -- j) {
          int sum=0;
          for (int i = 0; i <=j; ++i) {
              sum +=Math.pow(2,j-i)*t[i];
          }
          t[j] = sum;
      }
   }
}
```

(la fonction **Math.pow**(x,y) retourne x^y.)

1) Quelle est la complexité de l'algorithme **sommePre** (en fonction de n) ?
2) Faire tourner l'algorithme quand $n = 4$ et que le tableau t contient 1,0,0 et 1. Quelle sont alors les valeurs contenues dans t[0], t[1], t[2] et t[3] à la fin de l'algorithme ?
3) Dans le cas le plus général, que valent t[0], t[1], t[2], $\cdots$ t[n-1] à la fin de l'algorithme ?
4) Que peut-on faire avec un tel algorithme ?

E/- Piles (5 points).

On considère trois piles p_1, p_2 et p_3. La pile p_1 contient des nombres entiers strictement positifs tandis que p_2 et p_3 sont initialement vides. Dans tout ce qui suit, on utilisera les méthodes usuelles liées aux piles.

1) Ecrire un algorithme qui n'utilise que p_1, p_2 et p_3 et qui place tous les entiers de la pile p_1 dans la pile p_2 de façon à ce que les nombres pairs soient au dessus des nombres impairs.
2) Montrer sur une pile p_1 contenant 5 entiers dont 2 positifs et 3 négatifs comment fonctionne votre algorithme.
3) Ecrire le code Java correspondant à la question E/1) (on pourrait utiliser la classe **PileEntier** de la question C/3/).
4) On se propose de réaliser une calculatrice simplifiée évaluant les expressions en notation *postfixe* avec les opérateurs "+" et "−". Pour rappel, la notation postfixe de (2+5)−6+(4−2) est 2 5+6−4 2−+. On se donne une expression arithmétique ne contenant que des chiffres entre 0 et 9 et que les opérateurs binaires + et −. De plus on suppose que l'expression est donnée correctement. Ecrire un algorithme d'évaluation d'une expression postfixe (avec les hypothèses ci-dessus).
5) La classe String de Java dispose des méthodes `length()` renvoyant la longueur de la chaîne de caractères, `charAt(int i)` renvoyant le caractère en position i et `substring(int debut, int fin)` renvoyant le sous-mot formé des lettres entre la position `debut` et `fin-1`. Ecrire en Java l'algorithme de la question E/4).