Université Denis-Diderot
Licence I
Informatique: IF2

Examen Informatique IF2

Première session, durée 2 heures 30 minutes.
Tous les documents sont interdits.
(2 pages + QCM)

Cet examen comporte un questionnaire à choix multiples que vous devez rendre avec votre copie.
Pour vous identifier sur le questionnaire tout en conservant l'anonymat vous devez choisir un identifiant quelconque que vous mettrez en tête de votre copie et que vous reporterez sur le questionnaire et sur toutes les copies supplémentaires que vous allez rendre.
le barème est donné à titre indicatif. Le qcm est noté sur 10.

On considère une `PileInt` qui contient les méthodes: `public boolean estVide()`, `public int sommet()`, `public PileInt empiler(int v)`, `public PileInt depiler()`, `public void afficher()`. Pour l'instant on ne s'intéresse pas à l'implémentation de ces méthodes, mais on suppose que ces différentes méthodes satisfont les spécifications usuelles pour des piles d'entiers.

Dans les questions qui suivent exceptée la dernière, on ne s'autorise qu'à utiliser des variables entières ou booléennes simples, des variables `PileInt` et les méthodes définies dans `PileInt`. En particulier on ne pourra pas utiliser de tableaux ni d'autres structures de données.

1. (1pt) Ajouter à la classe `PileInt` une méthode d'instance qui affiche les éléments de la pile dans l'ordre inverse. (si $p = (X_1, ..., X_n)$ avec X_1 comme sommet de la pile, on affichera dans l'ordre X_n ... X_1).

   (Indication: On pourra utiliser une `PileInt` auxiliaire).

2. (3pts) Ajouter à la classe `PileInt` une méthode d'instance `concat(PileInt p)` qui ajoute la pile `p` en fin de la pile référencée par l'instance (si l'instance courante est $(X_1, \cdots, X_n)$ et $p = (Y_1, \cdots, Y_m)$ après `concat` l'instance courante sera $X_1, \cdots, X_n, Y_1, \cdots, Y_n$)) Ajouter aussi une méthode statique `concat(PileInt p1,PileInt p2)` qui ajoute `p2` au bout de `p1` et retourne `p1`.

   (Indication: on pourra donner une définition récursive de la concaténation de deux piles et traduire cette définition en un programme récursif).

   Quelle est en fonction de la hauteur de le pile `p1` (si $p = (X_1, \cdots X_n)$ la hauteur de p est n) et de la hauteur de la pile `p2` l'ordre de grandeur du nombre d'opérations effectuées par `concat`?

3. (2pts) Définir une méthode statique, `split(PileInt p, int v, PileInt min, PileInt max)` qui à partir de la pile `p`, construit les piles `min` et `max` telles que `min` contienne tous les éléments de `p` inférieurs à `v` et `max` contienne tous les éléments de `p` supérieurs ou égaux à `v`.

   Quelle est en fonction de la hauteur de le pile `p` (si $p = (X_1, \cdots X_n)$ la hauteur de p est n) l'ordre de grandeur du nombre d'opérations effectuées par `split` ?

4. (3pts) En s'inspirant du *tri-rapide* (*quicksort*) définir une méthode de tri de pile utilisant `split`. Quel est en fonction de la hauteur de la pile `p` (si $p = (X_1, \cdots X_n)$ la hauteur de p est n) l'ordre de grandeur du nombre d'opérations réalisées dans le pire cas et le meilleur cas pour cette méthode?

5. (2pts) On considère la classe `CelluleInt` qui permet de réaliser des listes d'entiers (`int`):

```
class CelluleInt {
    public int val;
    public CelluleInt suivant;
    public CelluleInt(int v, CelluleInt s) {
```

```
        val = v;
        suivant = s;
    }
}
```

Dans cette question on pourra utiliser toutes les structures de données voulues. Donner une implémentation des diverses méthodes de `PileInt` en utilisant la classe `CelluleInt` définie ci-dessus.

Comment pourrait-on procéder en utilisant une `interface`?