Composition d'Informatique

Les Principes des Langages de Programmation (INF 321)

Promotion 2005

Sujet proposé par Gilles Dowek

11 juillet 2006

Les exercices qui suivent sont indépendants et peuvent être traités dans n'importe quel ordre. On attachera une grande importance à la clarté, à la précision et à la concision de la rédaction.

Exercice 1 (4 points)

On définit la classe des listes

```
class List {
 int hd;
 List tl;

 List (int x, List y) {hd = x; tl = y;}}
```

1. Écrire une fonction `membre` qui prend en argument un entier `x` et une liste `l` et retourne un booléen qui indique si cet entier appartient à la liste ou non.

2. Écrire une fonction `supprime` qui prend en argument un entier `x` et une liste `l` et retourne une autre liste structurellement identique à `l` sauf que la première occurrence de l'entier `x` dans la liste `l`, si elle existe, a été supprimée.

3. Écrire une fonction `communs` qui prend en argument deux listes `l1` et `l2` et retourne le nombre d'éléments qui sont en commun dans les deux listes, en respectant les multiplicités (par exemple, le nombre d'éléments en commun dans les listes `1, 2, 2, 3, 4` et `2, 2, 2, 3, 5` est `3`).

4. Quelle est la complexité de cette fonction ? Comment faire pour trouver les éléments communs à deux listes d'entiers en `n ln n` et non en n^2 ? (Dans cette question on donnera l'idée sans écrire le programme.)

Exercice 2 (4 points)

On représente une file d'attente par un couple formée de deux listes, le couple formé des listes `l1` = $a_1, \ldots, a_n$ et `l2` = $b_1, \ldots, b_m$ représentant la file $a_1, \ldots, a_n, b_m, \ldots, b_1$ dont le premier élément est a_1 et le dernier b_1.

1. Que faire pour ajouter un élément à la fin de la file ? Quelle est la complexité de cette opération ?

2. Que faire pour accéder au premier élément de la file et le retirer, quand la liste `l1` est non vide ? Quelle est la complexité de cette opération ?

3. Proposer une solution pour accéder au premier élément de la file et le retirer, quand la liste `l1` est vide. Quelle est la complexité de cette opération ?

4. Montrer qu'il est possible de s'arranger pour que les `m - 1` prochains accès au premier élément de la file se fassent en temps constant. Quelle est donc la complexité moyenne de l'accès au premier élément de la file ?

Exercice 3 (3 points)

On considère le type suivant

```
class Arbre {
  int val;
  Arbre gauche;
  Arbre droite;

  Arbre (int x, Arbre y, Arbre z) {val = x; gauche = y; droite = z;}}
```

et le programme

```
Arbre a = new Arbre (0, new Arbre (2,null,null), new Arbre (1,null,null));
a.gauche.gauche = a.droite;
a.gauche.droite = a;
a.droite.gauche = a;
a.droite.droite = a.gauche;
```

1. Quels sont l'environnement et la mémoire construits ?
2. Représenter graphiquement cet état.
3. Quelle est la valeur de l'expression `a.gauche.gauche.droite.val` ?

Exercice 4 (4 points)

On considère le type des listes d'entiers de l'exercice 1.

1. Écrire une fonction qui prend en argument une liste d'entiers `l` et affiche toutes les paires d'entiers de la liste, en respectant les multiplicités. Par exemple, avec la liste `1, 2, 2, 5`, cette fonction affichera les six paires : {`1, 2`}, {`1, 2`}, {`1, 5`}, {`2, 2`}, {`2, 5`}, {`2, 5`}.

2. Écrire une fonction qui prend en argument une liste d'entiers `l` et affiche tous les nombres que l'on peut former en ajoutant, soustrayant, multipliant ou divisant deux nombres de cette liste. Dans cet exercice, tous les nombres sont des entiers naturels : la soustraction `x - y` est autorisée uniquement quand `x` $\geq$ `y` et la division `x / y` uniquement quand `x` est divisible par `y`.

3. Le *Compte est bon* est un jeu dans lequel un joueur se voit proposer une liste de nombres, par exemple `1, 10, 4, 4, 3, 50`, et un nombre qui n'est pas dans la liste, par exemple `688`, et il doit trouver un moyen d'atteindre ce nombre en utilisant les nombres de la liste et les quatre opérations par exemple `668 = ((1 + (50 - 3)) * (4 + 10)) - 4`. (Chaque nombre peut être utilisé autant de fois qu'il apparaît dans la liste. Comme à la question précédente,

la soustraction `x - y` est autorisée uniquement quand `x` $\geq$ `y` et la division `x / y` uniquement quand `x` est divisible par `y`.)

Écrire une fonction qui prend en argument une liste et affiche tous les nombres que l'on peut construire avec les nombres de cette liste et les quatre opérations.

Exercice 5 (4 points)

On ajoute au noyau impératif de Java décrit dans le chapitre 1 du poly un mécanisme d'exceptions simplifié formé d'une instruction `throw;` qui lève une exception.

La fonction Σ associe désormais à chaque triplet (formé d'un programme, d'un environnement et d'une mémoire) un couple formé d'un booléen qui vaut `normal` ou `exception` et d'une mémoire. La fonction Θ n'est pas modifiée (car dans ce cadre simple, les expressions ne contiennent pas d'appel de fonctions et ne peuvent pas lever d'exception).

1. Écrire la définition de la fonction Σ pour l'instruction `throw`.
2. Comment est modifiée la définition de la fonction Σ dans le cas de l'affectation ?
3. Comment est modifiée la définition de la fonction Σ dans le cas de la séquence ?
4. On ajoute au langage une construction `try p1 handle p2`. Définir la fonction Σ pour les instructions de cette forme.

Exercice 6 (1 point)

Pourquoi la déclaration de type

```
struct List {
 int hd;
 struct List tl;};
```

est-elle interdite en C alors que la déclaration

```
class List {
 int hd;
 List tl;
```

est autorisée en Java ?