

Corrigé de la Composition d'Informatique
Les Principes des Langages de Programmation (INF 321)

Promotion 2005

Sujet proposé par Gilles Dowek

11 juillet 2006

Exercice 1

1.

```
static boolean membre (int x, List l) {
    if (l == null) return false;
    if (x == l.hd) return true;
    return membre(x,l.tl);}
```

2.

```
static List supprime (int x, List l) {
    if (l == null) return l;
    if (x == l.hd) return l.tl;
    return new List (l.hd, supprime(x,l.tl));}
```

3.

```
static int communs (List l1, List l2) {
    if (l1 == null) return 0;
    if (membre(l1.hd,l2)) return 1 + communs(l1.tl,supprime(l1.hd,l2));
    return communs(l1.tl,supprime(l1.hd,l2));}
```

4. n^2 . Il faut commencer par trier les deux listes.

```
static int communs1 (List l1, List l2) {
    if (l1 == null) return 0;
    if (l2 == null) return 0;
    if (l1.hd == l2.hd) return 1 + communs1 (l1.tl,l2.tl);
    if (l1.hd < l2.hd) return communs1 (l1.tl,l2);
    return communs1 (l1,l2.tl);}
```

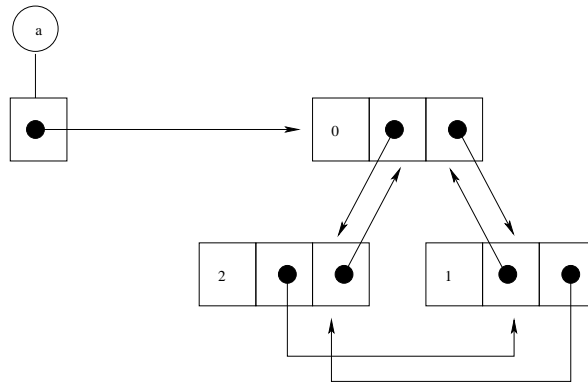
```
static int communs (List l1, List l2) {
    return(communs1(tri(l1),tri(l2)));}
```

Exercice 2

1. On ajoute l'élément en tête de la liste 12. Temps constant.
2. On lit le premier élément de la liste 11 et on le retire. Temps constant.
3. On retourne la liste 12 qui devient la nouvelle liste 11, puis on lit le premier élément de cette liste et on le retire. La complexité est proportionnelle à m .
4. La nouvelle liste 11 a la longueur $m - 1$, les $m - 1$ prochains accès au premier élément de la file se feront en temps constant. La complexité moyenne de ces m accès au premier élément de la liste est donc $(c \cdot m + d \cdot (m - 1)) / m$. L'accès au premier élément de la liste se fait donc en temps constant en moyenne.

Exercice 3

1. $e = [r1/a]$,
 $m = [r2/r1, (0, r3, r4)/r2, (2, r4, r2)/r3, (1, r2, r3)/r4]$
- 2.



3. 2.

Exercice 4

- 1.

```
static void paires1 (int v, List l) {
    if (l != null) {
        System.out.println(v + " " + l.hd);
        paires1(v, l.tl);
    }
}
```

```
static void paires (List l) {
    if (l != null) {
        paires1(l.hd, l.tl);
        paires (l.tl);
    }
}
```

- 2.

```

static void operations (int x, int y) {
    System.out.println (x + y);
    if (x >= y) System.out.println (x - y);
    if (y >= x) System.out.println (y - x);
    System.out.println (x * y);
    if (y != 0 && x % y == 0) System.out.println (x / y);
    if (x != 0 && y % x == 0) System.out.println (y / x);}

```

```

static void paires1 (int v, List l) {
    if (l != null) {
        operations(v,l.hd);
        paires1(v,l.tl);}}

```

```

static void paires (List l) {
    if (l != null) {
        paires1(l.hd, l.tl);
        paires (l.tl);}}

```

3.

```

static void resultat (int x, List r) {
    System.out.println(x);
    tous (new List(x,r));}

```

```

static void operations (int x, int y, List r) {
    resultat (x + y, r);
    if (x >= y) resultat (x - y, r);
    if (y >= x) resultat (y - x, r);
    resultat (x * y, r);
    if (y != 0 && x % y == 0) resultat (x / y,r);
    if (x != 0 && y % x == 0) resultat (y / x,r);}

```

```

static void paires1 (int v, List l, List r) {
    if (l != null) {
        operations(v,l.hd,List.revappend(l.tl,r));
        paires1(v,l.tl,new List (l.hd,r));}}

```

```

static void paires (List l, List r) {
    if (l != null) {
        paires1(l.hd, l.tl,r);
        paires (l.tl,new List(l.hd,r));}}

```

```

static void tous (List l) {
    paires (l,null);}

```

Exercice 5

1. $\Sigma(\text{throw } i, e, m) = (\text{exception}, w)$
2. $\Sigma(x := t, e, m) = (\text{normal}, m + (v/e(x)))$ où $v = \Theta(t, e, m)$
3. Si $\Sigma(p1, e, m) = (\text{normal}, m')$ alors $\Sigma(p1 \ p2, e, m) = \Sigma(p2, e, m')$
Et si $\Sigma(p1, e, m) = (\text{exception}, m')$ alors $\Sigma(p1 \ p2, e, m) = (\text{exception}, m')$
4. Si $\Sigma(p1, e, m) = (\text{normal}, m')$ alors $\Sigma(\text{try } p1 \ \text{handle } p2, e, m) = (\text{normal}, m')$
Et si $\Sigma(p1, e, m) = (\text{exception}, m')$ alors $\Sigma(\text{try } p1 \ \text{handle } p2, e, m) = \Sigma(p2, e, m')$

Exercice 6

Parce qu'en C la valeur d'un terme de type `struct List` est un couple et non une référence associée à un couple dans la mémoire, comme c'est le cas en Java.