

Corrigé de la Composition d'Informatique

Les Principes des Langages de Programmation (INF 321)

Promotion 2007

Sujet proposé par Gilles Dowek

7 juillet 2008

Exercice 1

```
static List succ (List l) {
    if (l == null) return new List(1,null);
    if (l.hd == 9) return new List (0,succ(l.tl));
    return new List (l.hd+1,l.tl);}

static List add (List l1, List l2, int c) {
    int d1,d2,s;
    List m1,m2;
    if ((l1 == null) && (l2 == null) && (c == 0)) return null;
    if (l1 == null) {d1 = 0; m1 = null;} else {d1 = l1.hd; m1 = l1.tl;}
    if (l2 == null) {d2 = 0; m2 = null;} else {d2 = l2.hd; m2 = l2.tl;}
    s = d1 + d2 + c;
    return new List (s%10, add(m1,m2,s/10));}
```

Exercice 2

oui. non. Par récurrence sur la longueur du mot. On distingue les cas où le mot est vide, de la forme $(_n w)_n$ et de la forme ww' .

```
class Word {
    boolean or;
    int ty;
    Word tl;

    Word (boolean x, int y, Word z) {or = x; ty = y; tl = z;}}

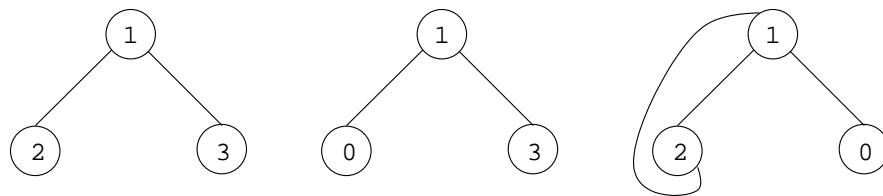
static boolean bienforme (Word w, Pile p) {
    if (w == null) return p.empty();
    if (w.or) {p.push (w.ty); return bienforme(w.tl,p);}
    if (p.empty ()) return false;
    if (p.top () == w.ty) {p.pop();return bienforme(w.tl,p);}
    return false;}
```

Exercice 3

```
static void croissante (int p, int m, List l, int n, int k) {
    if (n >= p) {List.printList(l);System.out.println();}
    else {croissante (p,m,new List(k,l),n+1,k);
          if (k < m) croissante (p,m,l,n,k+1);}
```

Soit $L(p, m)$ le nombre de listes croissantes de p entiers compris entre 0 et m . Une méthode, parmi d'autres, consiste à remarquer que les $L(p+1, m+1)$ listes croissantes de $p+1$ entiers compris entre 0 et $m+1$ se partitionnent entre celles qui commencent par 0 et qui sont en nombre $L(p, m+1)$ et celles qui commencent par un autre nombre et qui sont en nombre $L(p+1, m)$. On en déduit $L(p+1, m+1) = L(p+1, m) + L(p, m+1)$ et on conclut par une simple récurrence.

Exercice 4



false, true, elle ne termine pas. 3, 2, une infinité.

```
static boolean parcours2 (Arbre a, ListArbre l) {
    if (a == null) return false;
    if (ListArbre.mem (a,l)) return false;
    return (a.val == 0) || parcours2(a.gauche, new ListArbre(a,l)) ||
           parcours2(a.droite, new ListArbre(a,l));}
```

Exercice 5

