DEUG 1^{ere} année
MIAS 1 et MASS 1 – Programmation fonctionnelle

TD n° 6 – Corrigé

Exercice 1 :

Nbocc : *Objet, Liste* $\longrightarrow$ *Entier*

$$o, l \longmapsto \begin{cases} 0 & \text{si } l \text{ est vide} \\ Nbocc(o, l1) + Nbocc(o, l2) & \text{si } l = (l1 \ . \ l2) \\ 1 + Nbocc(o, l') & \text{si } l = (o \ . \ l') \\ Nbocc(l') & \text{si } l = (x \ . \ l') \text{ et } x \neq o \end{cases}$$

avec $l1 \in Liste$; $l2 \in Liste$; $o \notin Liste$; $x \notin Liste$.

```
(define Nbocc
   (lambda (o l)
      (cond ((null? l) 0)
            ((list? (car l)) (+ (Nbocc (car l)) (Nbocc (cdr l))))
            ((equal? o (car l)) (+ 1 (Nbocc (cdr l))))
            (else (Nbocc (cdr l))))))
```

Exercice 2 :

supnomb : *Liste* $\longrightarrow$ *Liste*

$$l \longmapsto \begin{cases} listevide & \text{si } l \text{ est vide} \\ (supnomb(l1) \ . \ supnomb(l2)) & \text{si } l = (l1 \ . \ l2) \\ supnomb(l1) & \text{si } l = (n \ . \ l1) \text{ et } n \text{ est un nombre} \\ (o \ . \ supnomb(l1)) & \text{si } l = (o \ . \ l1) \text{ et } o \text{ n'est pas un nombre} \end{cases}$$

avec $l1 \in Liste$; $l2 \in Liste$; $o \notin Liste$.

```
(define supnomb
   (lambda (l)
      (cond ((null? l) '())
            ((list? (car l)) (cons (supnomb (car l)) (supnomb (cdr l))))
            ((number? (car l)) (supnomb (cdr l)))
            (else (cons (car l) (supnomb (cdr l)))))))
```

Exercice 3 :

egalist? : *Liste, Liste* $\longrightarrow$ *Booléen*

$$l1, l2 \longmapsto \begin{cases} vrai & \text{si } l1 = l2 = listevide \\ faux & \text{si l'une des listes est vide et l'autre non} \\ egalist?(u, v) \wedge egalist?(l1', l2') & \text{si } l1 = (u \ . \ l1'); l2 = (v \ . \ l2') \\ & \text{et } u \in Liste; v \in Liste \\ egalist?(l1', l2') & \text{si } l1 = (x \ . \ l1'); l2 = (y \ . \ l2') \\ & \text{et } x \text{ et } y \text{ sont des atomes et } x = y \\ faux & \text{dans tous les autres cas} \end{cases}$$

```
(define egalist?
   (lambda (l1 l2)
      (cond ((and (null? l1) (null? l2))
             #t)
            ((or (null? l1) (null? l2))
             #f)
            ((and (list? (car l1)) (list? (car l2)))
             (and (egalist? (car l1) (car l2))
                  (egalist? (cdr l1) (cdr l2))) )
            ((and (atom? (car l1)) (atom? (car l2)) (equal? (car l1) (car l2)))
             (egalist? (cdr l1) (cdr l2)))
            (else #f))))
```