Université Bordeaux 1 Licence S&TS Mention Informatique S5 2012/2013

# UE INF5011 Programmation 3

## Programmation Fonctionnelle et Symbolique

### Devoir surveillé

Tous documents autorisés. Mardi 16 octobre 2012 Durée : 1h20.
Le barème est donné à titre **indicatif**.

## Exercice 1 *(3pts)*

Évaluer les expressions suivantes :

1. `(cons nil (cons '(1 2) nil))`
2. `(list nil '(1 2) nil)`
3. `(append nil '(1 2) nil)`
4. `(last '(1 2 3))`
5. `(caddr '(1 2 3))`
6. `(mapcar #'* '(1 3 5) '(2 4 6) '(1 1 1))`

## Exercice 2 *(6pts)*

1. Écrire une fonction `at-least-one (l preds)` qui prend en paramètre une liste d'éléments `l` et une liste de prédicats `preds` applicables aux éléments de `l` et qui retourne la liste des éléments de `l` qui vérifient **au moins** un des prédicats de `preds`. Exemples :

```
CL-USER> (at-least-one '(1 2 3 4 5 6) '())
NIL
CL-USER> (at-least-one '(1 2 3 4 5 6) (list #'evenp))
(2 4 6)
CL-USER> (at-least-one '(1 2 3 4 5 6) (list #'evenp (lambda (x) (>= x 5))))
(2 4 5 6)
```

2. Modifier votre fonction de manière à ce qu'à l'appel, les prédicats ne soient plus donnés dans une liste mais sous forme d'arguments en nombre variable. Exemple :

```
CL-USER> (at-least-one '(1 2 3 4 5 6) #'evenp (lambda (x) (>= x 5)))
(2 4 5 6)
```

## Exercice 3 *(5pts)*

1. Écrire une fonction `nb-cons (e)` qui retourne le nombre de paires constituant l'objet représenté par l'expression `e`.

Exemples :

```
CL-USER> (nb-cons '())
0
CL-USER> (nb-cons 1)
0
CL-USER> (nb-cons '(1 . 2))
1
CL-USER> (nb-cons '((1 2) (2 3)))
6
```

2. La fonction écrite est-elle récursive terminale ?

## Exercice 4 *(6pts)*

Écrire des expressions utilisant uniquement des atomes et la fonction `cons` telles que le `Printer` affiche les résultats suivants :

1. `(1 . 2)`
2. `(1 2)`
3. `((1 . 2) (3 . 4))`
4. `((1 . 2) 3 . 4)`

Soit la fonction `pair-list (e)` suivante :

```
(defun pair-list (e)
  (if (atom e)
      e
      (let ((car (pair-list (car e)))
            (cdr (pair-list (cdr e))))
        (cons car
              (if (listp cdr)
                  cdr
                  (list cdr))))))
```

Évaluer les expresssions suivantes :

5. `(pair-list '())`
6. `(pair-list 1)`
7. `(pair-list '(1 . 2))`
8. `(pair-list '((1 . 2) (3 . 4)))`

FIN