Université Bordeaux 1 Licence S&T Mention Informatique S5 2007/2008

UE INF353 **Programmation 3**

Programmation Fonctionnelle et Symbolique
Devoir surveillé No 1

Tous documents autorisés. Vendredi 19 Octobre 2007
Durée : 1h.
Le barême est donné à titre **indicatif**.
Chacun des trois exercices doit être **rédigé** sur une feuille **séparée** sur laquelle doit être inscrit votre nom et votre numéro de groupe. Vous devez rendre exactement **une feuille** (ou copie), éventuellement blanche, **par exercice**.

Dans les exercices 2 et 3 il est **obligatoire** *de programmer dans un style* **fonctionnel**, *c'est-à-dire sans utiliser ni* `setf` *ni boucles.*

Exercice 1 *(4pts)*
Évaluer les expressions suivantes :

1. `(cons '(1 2 3) 4)`
2. `(cons '(1 2 3) ())`
3. `(list '(1 2 3) '(a b) ())`
4. `(append '(1 2 3) '(4 5) ())`
5. `(last '(1 2 3 4))`
6. `(butlast '(1 2 3 4))`
7. `(sort '(2 5 7 3 6) #'<)`
8. `(sort '((1 . 2) (5 . 5) (2 . 7) (4 . 3) (7 . 6)) #'< :key #'cdr)`

Exercice 2 *(9pts)*
On considère deux ensembles de symboles S_1 et S_2 ne contenant pas le symbole `NIL`. On définit un *traducteur* de symboles comme
une **fonction** de $S1$ vers $S_2 \cup$ `NIL` qui à $s \in S_1$ associe un symbole de S_2 (sa *traduction*) ou `NIL` si la traduction n'est pas définie.
Soit la fonction `francais-anglais` suivante :

```
(defun francais-anglais ()
  (lambda (s)
    (case s
      (bleu 'blue)
      (rouge 'red)
      (jaune 'yellow))))
```

1. Quel type d'objet `Lisp` retourne l'appel `(francais-anglais)` ?
2. L'objet retourné par l'appel `(francais-anglais)` est-il un traducteur (comme défini ci-dessus) ?

3. Écrire la fonction `(traduire symbole traducteur)` qui retourne la traduction du symbole `symbole` par le traducteur `traducteur` si elle réussit et le symbole `???` sinon.

```
CL-USER> (traduire 'rouge (francais-anglais))
RED
CL-USER> (traduire 'vert (francais-anglais))
???
```

4. Écrire une fonction `(traduire-liste symboles traducteur)` qui retourne la liste des traductions des symboles de la liste `symboles` avec le traducteur `traducteur`. Exemple :

```
CL-USER> (traduire-liste  '(rouge bleu vert jaune rouge) (francais-anglais))
(RED BLUE ??? YELLOW RED)
```

5. Écrire une fonction `(traducteur l1 l2)` qui, étant données deux listes de symboles `l1` et `l2` de longueurs identiques retourne un traducteur qui traduit le $i$ème symbole `l1` en le $i$ème symbole de `l2` pour tout i compris entre 1 et la longueur des listes. On pourra s'aider de la fonction `pairlis`, vue en cours et dont la description `Hyperspec` est donnée en annexe. Exemple d'utilisation :

```
(defvar *avocat*
  (traducteur '(k l m n o p q r s t u v w x y z a b c d e f g h i j)
              '(a b c d e f g h i j k l m n o p q r s t u v w x y z))
  "traducteur du code 'A vaut K'")

CL-USER> (traduire-liste '(a b k 4 c k) *avocat*)
(Q R A ??? S A)
```

Exercice 3 *(7pts)*

Soit la fonction `(distribute e lists)` qui s'applique à un élément `e` et à une liste de listes `lists` et qui retourne la liste de ces listes avec en plus l'élément `e` en tête de liste.
Exemple :

```
CL-USER> (distribute 3 '((1) (2 2) (3 3 3) (4 4 4 4)))
((3 1) (3 2 2) (3 3 3 3) (3 4 4 4 4))
```

Écrire deux versions de la fonction `distribute` :

1. une version récursive,
2. une version utilisant la fonction `mapcar`.

Annexe : documentation `Hyperspec` des fonctions `sort` et `pairlis`.

FIN