Examen en "Programmation Applicative" - FLIN304
L2 Informatique

Session 2, année 2008-2009, juin 2009.

Durée : 2h. Tous documents autorisés. Barême indicatif pouvant être modulé selon les résultats. Sections indépendantes.

1 Bases : valeurs d'expressions, portée des identificateurs, listes - 4 points

On évalue les expressions ci-dessous dans l'ordre indiqué. Donnez la valeur de chaque expression. Certaines expressions n'ont pas de valeur Certaines évaluations provoquent une erreur. Dans ce cas vous indiquerez la cause de l'erreur.

```
(define x 4)
(define y (+ x 2))
(define f (lambda (x) (let ((y (+ x 1))) (+ x y))))
(+ x (f 2))
(and (< x 5) (> y 5))
(define x 8)
y
'(+ 2 (+ x 1))

(define li '(1 2 3))
(eq? li '(1 2 3))
(cons li (list 4 5))
(append li (list 4 5))
```

2 Fonctions sur les listes (6 points)

Questions

1. Ecrire la fonction `compter` qui prend un symbole-lettre et une liste de symboles-lettres et compte les occurrences de ce premier parmi ces derniers. Exemple :

   ```
   (compter 'N '(A N T I C O N S T I T U T I O N N E L L E M E N T))
   = 5
   ```

2. Écrire la fonction `begaie` prenant une liste de symboles en argument représentant une phrase, et retournant en sortie une liste de mots où tous les mots sont répétés. Par exemple :

   ```
   (begaie '(COMMENT ALLEZ VOUS))
   = (COMMENT COMMENT ALLEZ ALLEZ VOUS VOUS)
   ```

3. Écrire une fonction `debegaie` qui ôte d'une phrase tout bégaiement et notamment celui produit par la fonction de l'exercice précédent. Par exemple :

   ```
   ? (DEBEGAIE (BEGAIE '(COMMENT ALLEZ VOUS)))
   = (COMMENT ALLEZ VOUS)
   ```

   Aide : si une liste n'a qu'un élément, il n'y a rien à faire, de plus, on ne peut comparer le `car` et le `cadr` d'une liste que si elle contient au moins deux termes.

4. Donnez une version récursive terminale de `compter`.
5. Moins simple (2 points), donnez une version récursive terminale de `begaie`.

3 Fonctions sur les nombres - Sur 5 points

1. Définissez une fonction récursive non terminale, nommée `pow2` qui calcule 2^n (N'utilisez évidemment pas la fonction puissance de *scheme* `expt`).

2. Définissez une fonction récursive terminale `iter-pow2` calculant la même chose.
3. Définissez une fonction `powerof2?` rendant un booléan disant si un nombre passé en argument est ou non une puissance de 2. On rappelle que les fonctions *scheme* `remainder` et `quotient` calculent respectivement le reste et le quotient de la division entière.

```
(powerof2? 16)
= #t
(powerof2? 17)
= #f
(powerof2? 0)
= #f
(powerof2? 1)
= #t
```

4. Définissez la même fonction en utilisant aucun `if` ni `cond`, mais uniquement des `and` et `or`.
5. Généralisez en définissant une fonction à deux paramètres n et p disant si n est une puissance de p.

4 Problème - 5 points

On dispose d'une liste (ou liste d'association ou dictionnaire) de produits, chaque produit est représenté par une **liste** de 2 éléments représentant son nom et son prix : `(nom prix)`, par exemple `(livre 10)`.

1. Définissez les fonctions `nom` et `valeur` donnant respectivement le nom et le prix d'un produit.

```
(define produit1 '(balle 10))
(nom produit1)
= balle
(valeur produit1)
= 10
```

2. Quel est l'intérêt des deux fonctions précédentes d'un point de vue "abstraction de données".
3. Dans les questions suivantes on supposera que tous les produits ont un nom et un prix différent.
   Définissez la fonction `(prix nom lp)` qui donne le prix du produit de nom `nom` dans la liste de produits `lp`.

```
(define Lprod '((voiture 40) (peluche 50) (velo 100) (balle 10) (train
60) (cube 11)(disque 12))

(prix 'velo Lprod)
= 100
```

4. Définissez la fonction `(plusCher lp)` qui rend le produit le plus cher de la liste de produits `lp`. Par exemple `(plusCher Lprod)` vaut `(vélo 100)`.
5. On s'intéresse maintenant au problème suivant : étant donné une liste de produits lp et un crédit `s`, calculer une sous-liste des produits qui peuvent être achetés avec ce crédit en prenant obligatoirement en premier, à chaque itération, le plus cher. Si la somme est inférieure au produit le plus cher, la sous-liste rendue est vide.
   Pour cela on peut sélectionner en premier le produit le plus cher, si son prix est inférieur ou égal au crédit, on le place dans la liste d'achat et on continue avec le reste des produits et le crédit amputé du prix du produit sélectionné.
   Ecriver la fonction `achatCher` traitant ce problème.
   Exemple :

```
(achatCher Lprod 90)
=()
(achatCher Lprod 210)
=((velo 100) (train 60) (peluche 50))   ;; l'ordre des produits est indifférent
```