Université Bordeaux 1 Licence S&TS Mention Informatique S5
2009/2010

UE INF353 Programmation 3

Programmation Fonctionnelle et Symbolique

Devoir surveillé No 2

Tous documents autorisés. Mercredi 2 Décembre 2009 Durée : 1h20.
Le barème est donné à titre **indicatif**.
Sur chaque feuille, indiquez vos nom, prénom et numéro de groupe.

Exercice 1 *(5pts)*

Soit la fonction `mystere-p (n)` donnée Figure 1.

```
(defun mystere-p (n)
  (assert (integerp n))
  (when (or (<= n 1) (and (> n 2) (evenp n)))
    (return-from mystere-p nil))
  (loop
     for d from 3 to (sqrt n) by 2
     when (zerop (mod n d))
     do (return-from mystere-p nil)
     finally (return t)))
```

FIG. 1 – Fonction `mystere-p` de l'exercice 1

1. Que retourne l'appel `(mystere-p 3)` ?
2. Que retourne l'appel `(mystere-p 4)` ?
3. Que retourne l'appel `(mystere-p 9)` ?
4. Expliquer en une phrase ce que fait la fonction `mystere-p`.
5. Donner une nouvelle version de cette fonction en remplaçant la boucle `loop` par une boucle `do`.

Exercice 2 *(5pts)*

Soit la fonction `mystere` donnée Figure 2.

```
(defun mystere (tab)
  (let ((len (length tab)))
    (if (zerop len)
      0
      (do ((i 1 (1+ i))
           (v (aref tab 0))
           (changements 1))
          ((>= i len) changements)
        (unless (= v (aref tab i))
          (incf changements)
          (setf v (aref tab i)))))))
```

FIG. 2 – Fonction `mystere` de l'exercice 2

1. Que retourne l'appel `(mystere #())` ?
2. Que retourne l'appel `(mystere #(1 2 3))` ?
3. Que retourne l'appel `(mystere #(1 1 2 3))` ?
4. Expliquer en une phrase ce que fait la fonction `mystere`.
5. Donner une nouvelle version de cette fonction en remplaçant la boucle `do` par une boucle `loop`.

Exercice 3 *(5pts)*

On considère des tableaux contenant des entiers **positifs ou nuls**.

1. Écrire une fonction **itérative** `max-tab (tab)` qui retourne l'élément maximal contenu dans le tableau `tab` ou 0 si le tableau est vide. Exemples

   ```
   CL-USER> (max-tab #(1 2 3 5 2 5 1 5 1 5))
   5
   ```

2. Écrire une fonction **itérative** `frequence (tab)` dont l'argument `tab` est un tableau non vide d'entiers positifs ou nuls, et qui retourne le tableau des fréquences des valeurs comprises entre 0 et la valeur maximum contenue dans `tab`. Exemple :

   ```
   CL-USER> (frequence #(1 2 3 5 2 5 1 5 1 5))
   #(0 3 2 1 0 4)
   ```

Exercice 4 *(5pts)*

Dans cet exercice on pourra utiliser indifféremment récursion ou itération.
On considère ses mots représentés par des symboles Lisp.
On représente un ensemble de phrases contenant des mots par un graphe acyclique constitué de tables de hachages.

– Les clés des tables sont des mots ;
– les valeurs sont soit `NIL`, soit une table contenant la suite des phrases possibles.

Les mots seront simplement représentés par des symboles Lisp.

1. Donner une expression qui construit la table (toi, moi, lui) de l'exemple et qui la mémorise dans une variable `*t1*`.
2. Donner une expression qui construit la table (a) de l'exemple et qui la mémorise dans une variable `*t2*`.
3. Écrire une fonction `existe (phrase)` qui retourne vrai si la phrase appartient au corpus, faux sinon. Exemples :

```
CL-USER> (existe-phrase '(il pense a lui toujours) *corpus*)
NIL
CL-USER> (existe-phrase '(il pense a lui) *corpus*)
T
```

4. Écrire une fonction `corpus-to-list (corpus)` qui retourne la listes des phrases du corpus, chaque phrase étant représentée par une liste.

```
CL-USER> (corpus-to-list nil)
(NIL)
CL-USER> (corpus-to-list *t1*)
((LUI) (MOI) (TOI))
CL-USER> (corpus-to-list *t2*)
((A TOI) (A MOI) (A LUI))
CL-USER> (corpus-to-list *corpus*)
((IL PENSE A TOI) (IL PENSE A MOI) (IL PENSE A LUI) (LE JOUR VIENDRA)
 (LE CHAT MANGE) (LE CHAT DORT) (UN JOUR VIENDRA) (UN CHAT MANGE)
 (UN CHAT DORT))
CL-USER> (corpus-to-list nil)
(NIL)
```

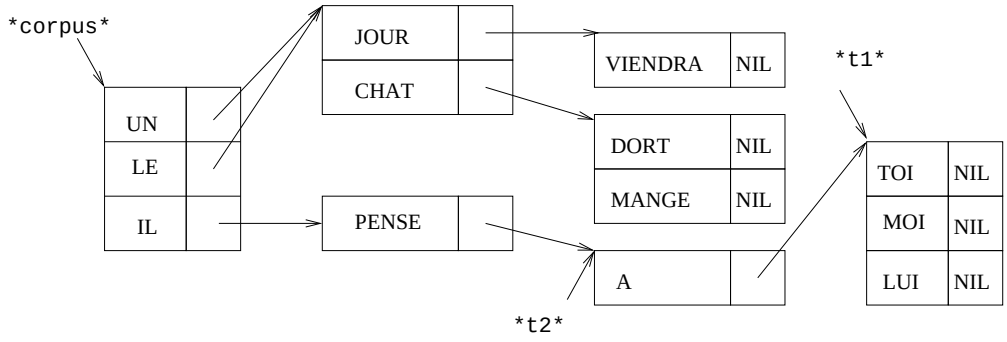

FIG. 3 – Ensemble de phrases représenté par un graphe acyclique

FIN