UE INF5011 Programmation 3

Programmation Fonctionnelle et Symbolique

Devoir surveillé

Tous documents autorisés. Mercredi 16 octobre 2013 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 1 '(nil 2) nil)`
3. `(append '(nil) '(1 2) nil)`
4. (last '((1) (2 3)))
5. `(cdadr '((1) (2 3)))`
6. `(mapcan (lambda (x) (if (> x 3) '() (list x))) '(1 3 5 2 4 6))`

On représente un ensemble $E \subseteq \mathbb{N}$, fini ou infini, par sa fonction caractéristique f_E :

$$\begin{array}{rccl} f_E : & \mathbb{N} & \longrightarrow & \mathbb{B} \\ & i & \mapsto & f_E(i) \quad \textit{avec } f_E(i) \Longleftrightarrow i \in E \end{array}$$

Exercice 2 *(9pts)*

Soit le début d'implémentation donné par la figure **??** en annexe.

7. Quel est le type de la valeur de la variable `*i-evens*` ?
8. Évaluer les expressions suivantes :
   (a) `(i-member 3 *i-empty*)`
   (b) `(i-member 3 *i-evens*)`
9. Définir une variable `*i-naturals*` représentant $\mathbb{N}$.
10. Écrire une fonction `multiple-of (n)` qui retourne l'ensemble des entiers multiples de `n`.
11. Écrire une fonction `i-complement (set)` qui retourne le complémentaire de `set` dans $\mathbb{N}$.
12. Écrire une fonction `i-intersection (set1 set2)` qui retourne l'intersection des ensembles `set1` et `set2`.

Exemples :

```
CL-USER> (i-member 3 *i-naturals*)
T
CL-USER> (defparameter *s* (i-intersection *i-evens* (multiple-of 3)))
*S*
CL-USER> (i-member 12 *s*)
T
CL-USER> (i-member 12 (i-complement *s*))
NIL
```

13. Modifier votre fonction `intersection` de manière à ce qu'elle accepte un nombre quelconque d'arguments. Exemple :

```
CL-USER> (i-member
          77
          (i-intersection
           *i-odds*
           (i-complement (multiple-of 3))
           (i-complement (multiple-of 5))))
T
```

Exercice 3 *(8pts)*

14. Écrire une fonction `i-less-than (set n)` qui retourne la liste ordonnée des éléments de l'ensemble `set` qui sont plus petits que l'entier `n`.
15. La fonction écrite est-elle récursive terminale ?

Exemples :

```
CL-USER> (defparameter *s* (i-intersection *i-evens* (multiple-of 3)))
*S*
CL-USER> (i-less-than *s* 20)
(0 6 12 18)
CL-USER> (i-less-than (i-complement *s*) 10)
(1 2 3 4 5 7 8 9 10)
```

16. Écrire une fonction `i-sequence (set n)` qui retourne la liste ordonnée des `n` premiers éléments de l'ensemble `set`.
17. La fonction écrite est-elle récursive terminale ?

Exemples :

```
CL-USER> (i-sequence *i-naturals* 10)
(0 1 2 3 4 5 6 7 8 9)
CL-USER> (i-sequence *s* 10)
(0 6 12 18 24 30 36 42 48 54)
```

18. Rajouter un paramètre mot-clé `start` à la fonction `i-sequence` de manière à ce qu'elle retourne toujours `n` éléments mais en commençant à partir de l'élément de rang `start` qui sera par défaut `0` (rang correspondant au premier élément). Exemples :

```
CL-USER> (i-sequence *i-naturals* 10 :start 4)
(4 5 6 7 8 9 10 11 12 13)
CL-USER> (i-sequence *i-odds* 10 :start 4)
(5 7 9 11 13 15 17 19 21 23)
```

FIN

Annexe

```
(defun i-member (i set)
  "does I belong to the set SET"
  (funcall set i))

(defvar *i-empty*
  (lambda (i)
    (declare (ignore i))
    nil)
  "empty set")

(defvar *i-evens*
  (lambda (i)
    (evenp i))
  "set of even natural integers")

(defvar *i-odds*
  (lambda (i)
    (oddp i))
  "set of odd natural integers")

(defun i-union (s1 s2)
  "union of S1 and S2"
  (lambda (i)
    (or (i-member i s1) (i-member i s2))))
```

FIG. 1 – Implémentation pour les ensembles d'entiers