Université Bordeaux 1 Licence S&T Mention Informatique S6 2007/2008

UE INF 353T- Programmation Fonctionnelle et Symbolique
Contrôle continu 2

Tous documents autorisés. Mardi 25 Mars 2008
Durée : 1h.
Vous devez écrire vos solutions dans un fichier cc2.lisp et envoyer ce fichier à l'adresse : `ges@labri.fr` avec comme sujet "CCPFS".

Exercice 1 *(20pts)*

1- On souhaite écrire une fonction `lmultiples(d inf sup)` qui retourne la liste des entiers n qui sont multiples de `d` et tels que `inf` $\leq n \leq$ `sup`. Par exemple, on doit avoir

```
CL-USER> (lmultiples 3  5 14)
(6 9 12)
```

1.1- Écrire une version `lmultiplesdo(d inf sup)`, qui utilise la boucle do.
1.2- Écrire une version `lmultiplesloop(d inf sup)`, qui utilise la boucle loop.

On cherche à implémenter la notion d'ensemble d'entiers par des listes dont le premier élément n'est pas un entier (on appelle cet élément une *sentinelle*). Par exemple, si la valeur de la sentinelle est fixée à nil, l'ensemble $\{2, 3, 4\}$ est représenté par la liste `(nil 2 3 4)`.
Le fichier `http://dept-info.labri.u-bordeaux.fr/~ges/ENSEIGNEMENT/PFS/initial-cc2.lisp` contient quelques constantes et fonctions de base sur cette représentation des ensembles d'entiers.
2- Écrire une fonction `nset-adjoin(e s)` qui modifie l'ensemble s en lui ajoutant l'élément e et retourne t (resp nil) suivant que e appartenait déjà à e (ou non). Quelle(s) différence(s) y-a-t-il entre `set-adjoin(e s)` (fournie dans le fichier ci-dessus) et votre fonction `nset-adjoin(e s)` ?
3- Ecrire une fonction `smultiples(d inf sup)` qui renvoie un ensemble dont les éléments sont ceux de la liste `lmultiples(d inf sup)`.
4- On veut écrire une fonction `union(s1 s2)` qui renvoie la réunion des ensembles $s1, s2$ (sans les modifier).
4.1- Écrire une version `uniondo(s1 s2)` qui utilise la boucle do.
4.2- Écrire une version `unionloop(s1 s2)` qui utilise la boucle loop.
5- On veut écrire une fonction `intersection(s1 s2)` qui renvoie l'intersection des ensembles $s1, s2$ (sans les modifier).
5.1- Écrire une version `intersectiondo(s1 s2)` qui utilise la boucle do.
5.2- Écrire une version `intersectionloop(s1 s2)` qui utilise la boucle loop.
6- Tester vos implémentations des opérations union et intersection sur les ensembles calculés par la fonction `smultiples`. Par exemple, on doit avoir

```
CL-USER> (set-intersectiondo (smultiples 2 0 100) (smultiples 3 0 100))
(NIL 0 6 12 18 24 30 36 42 48 54 60 66 72 78 84 90 96)
```