
	<p style="text-align: center;">ANNÉE UNIVERSITAIRE 2010/2011 2IÈME SESSION D'AUTOMNE</p> <p><b>Parcours :</b> CSB5  <b>Code UE :</b> INF353  <b>Épreuve :</b> Programmation 3  <b>Date :</b> Vendredi 09 juin 2011  <b>Heure :</b> 8h30  <b>Durée :</b> 1h30  Documents : autorisés  Épreuve de Mme Irène Durand</p>	
---	--	---

Le barème est donné à titre **indicatif**.

Le sujet comporte ?? pages.

**Exercice 1** (2pts)

Évaluer les expressions suivantes :

1. (cons '() '(1))
2. (list '() '(1))
3. (append '() '(1))
4. (cons '() 1)

**Exercice 2** (3pts)

Soit le programme :

```
(defparameter *p* 2)
(let ((i -1))
  (defun next ()
    (setf i (mod (1+ i) *p*)))
  (defun set-mod (n)
    (setf *p* n))
  (defun set-i (n)
    (setf i n)))
```

Compléter le scénario suivant :

```
CL-USER> (next)
;; Réponse 1
CL-USER> (next)
;; Réponse 2
CL-USER> (next)
;; Réponse 3
CL-USER> (set-mod 3)
;; Réponse 4
CL-USER> (set-i 2)
;; Réponse 5
CL-USER> (next)
;; Réponse 6
```

### Exercice 3 (4pts)

Écrire une macro `zero-if` (`expr zero-expr not-zero-expr`) qui réalise une variante de l'instruction `if`. Le premier argument `expr` doit être une expression numérique tandis que les arguments `zero` et `not-zero` sont des expressions quelconques.

La sémantique de l'instruction `zero-if` est la suivante : si le résultat de l'évaluation de `expr` n'est pas un nombre alors la valeur `NIL` est retournée, sinon si le résultat est égal à 0 alors l'expression `zero-expr` est évaluée, sinon c'est l'expression `not-zero-expr` qui est évaluée.

Exemples :

```
CL-USER> (zero-if (list 2 2) 'zero 'not-zero)
NIL
CL-USER> (zero-if (- 2 2) 'zero 'not-zero)
ZERO
CL-USER> (zero-if (+ 2 2) 'zero 'not-zero)
NOT-ZERO
```

### Exercice 4 (11pts)

On souhaite implémenter des **listes** d'éléments, **doublément chaînées** et **avec sentinelles** pour éviter les cas particuliers quand on effectue des opérations en début et en fin de liste. Nous appellerons une telle liste *dliste*. Un exemple schématique est présenté Figure ??.

Chaque cellule (`cell`) d'une dliste aura trois créneaux `elem`, `prev` et `next` contenant respectivement l'élément, la cellule précédente, la cellule suivante.

Une dliste (`dlist`) aura deux créneaux `start` et `end` contenant respectivement la sentinelle de début et la sentinelle de fin de liste.

Soient les opérations définies page ??.

1. Implémenter la classe des cellules (`cell`).
2. Implémenter la classe des dlistes (`dlist`).
3. Définir la fonction `make-empty-dlist` () qui retourne une nouvelle dliste vide.
4. Implémenter l'opération `dlist-empty-p` (`dlist`).
5. Implémenter l'opération `dadd-start` (`e dlist`).
6. Implémenter l'opération `dadd-end` (`e dlist`).
7. Compléter l'implémentation de l'opération `print-object` de manière à ce que les cellules s'affichent comme dans les exemples qui suivent.
8. Implémenter une version spécialisée de l'opération `print-object` pour les dlistes de sorte qu'une dliste vide s'affiche [] et une dliste contenant les éléments 1,2,3 s'affiche [ 1 2 3].
9. Implémenter l'opération `dmapcar` (`f dlist`).

Exemples :

```
CL-USER> (defparameter *dl* (make-empty-dlist))
*DL*
CL-USER> *dl*
```

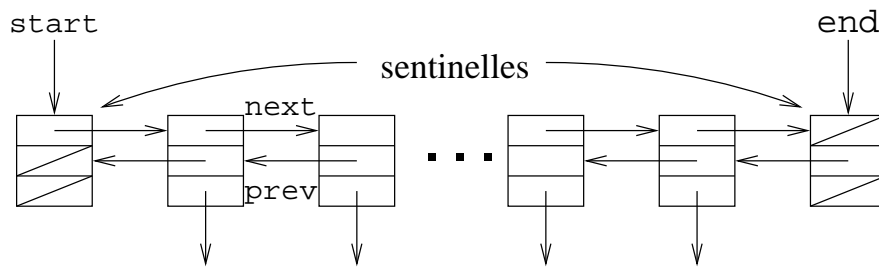


FIGURE 1 – Liste doublement chaînée avec sentinelles

```

[]
CL-USER> (dlist-empty-p *dl*)
T
CL-USER> (dadd-start 1 *dl*)
[ 1]
CL-USER> (dlist-empty-p *dl*)
NIL
CL-USER> (dadd-end 2 *dl*)
[ 1 2]
CL-USER> (dadd-end 3 *dl*)
[ 1 2 3]
CL-USER> (dfirst-cell *dl*)
#<CELL {10056D95D1}> 1
CL-USER> (dlast-cell *dl*)
#<CELL {10057F35D1}> 3
CL-USER> (next (dfirst-cell *dl*))
#<CELL {10056D9581}> 2
CL-USER> (prev (dlast-cell *dl*))
#<CELL {10056D9581}> 2
CL-USER> (dlast-elem *dl*)
3
CL-USER> (elem (dfirst-cell *dl*))
1
CL-USER> (dmapcar (lambda (x) (* 2 x)) *dl*)
[ 2 4 6]
CL-USER> *dl*
[ 1 2 3]

```

```

(defgeneric dlist-empty-p (dlist)
  (:documentation "prédicat qui teste si DLIST est vide"))

(defgeneric start (dlist)
  (:documentation "retourne la sentinelle de début de dlist"))

(defgeneric end (dlist)
  (:documentation "retourne la sentinelle de fin de DLIST"))

(defgeneric dfirst-cell (dlist)
  (:documentation
   "retourne la cellule contenant le premier élément de DLIST;
   nécessite que dlist soit non vide"))

(defgeneric dlast-cell (dlist)
  (:documentation
   "retourne la cellule contenant le dernier élément de DLIST;
   nécessite que DLIST soit non vide"))

(defgeneric dfirst-elem (dlist)
  (:documentation
   "retourne le premier élément de DLIST;
   nécessite que DLIST soit non vide"))

(defgeneric dlast-elem (dlist)
  (:documentation
   "retourne le dernier élément de DLIST
   nécessite que DLIST soit non vide"))

(defgeneric dadd-start (e dlist)
  (:documentation "ajoute l'élément E au début de DLIST"))

(defgeneric dadd-end (e dlist)
  (:documentation "ajoute l'élément E à la fin de DLIST"))

(defgeneric dmapcar (f dlist)
  (:documentation
   "retourne une nouvelle dliste dont les éléments sont
   ceux de DLIST auxquels on a appliqué la fonction F"))

```

FIN