
	<p>ANNÉE UNIVERSITAIRE 2008/2009 1ÈRE SESSION D'AUTOMNE</p> <p>Parcours : CSB5 Code UE : INF353 Épreuve : Programmation Fonctionnelle et Symbolique Date : Jeudi 18 décembre 2008 Heure : 8h30 Durée : 1h30 Documents : autorisés Épreuve de Mme Irène Durand</p>	
---	--	---

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

Le sujet comporte 4 pages.

Exercice 1 (2pts)

Soit le bout de code suivant

```
(let ((y 10))
  (defun fstat (x)
    (+ y x)))
```

```
(defun gstat (y)
  (fstat y))
```

```
(defvar *x* 0)
```

```
(let ((*x* 10))
  (defun fdyn (x)
    (+ *x* x)))
```

```
(defun gdyn (*x*)
  (fdyn *x*))
```

1. Que retourne l'appel (gstat 5) ?

2. Que retourne l'appel (gdyn 5) ?

Exercice 2 (2pts)

1. Écrire une fonction `finverse` (`f`) qui étant donnée une fonction numérique $f : \mathbb{R} \rightarrow \mathbb{R}$ retourne la fonction

$$\begin{aligned} \text{finverse}(f) : \mathbb{R} &\rightarrow \mathbb{R} \\ x &\mapsto 1/f(x) \end{aligned}$$

2. Donner un exemple d'appel à une fonction retournée par la fonction `finverse`.

Exercice 3 (9pts)

1. Écrire une fonction `moyenne` qui retourne la moyenne des valeurs contenues dans une liste. Exemple :

```
CL-USER> (moyenne '(12 14 11 10 8 15))
35/3
```

Pour un semestre universitaire d'une filière donnée, on dispose de la liste de ses unités d'enseignement. Exemple :

```
CL-USER> *s5*
(INF351 INF353 INF355 INF157 MHT411 LGV304)
```

Pour chaque étudiant inscrit dans ce semestre, on dispose, sous forme de liste d'association, la liste des notes qu'il a obtenues aux unités d'enseignement qu'il a présentées.

```
CL-USER> *notes-etudiant1*
((INF351 . 12) (INF353 . 15) (INF355 . 12) (INF157 . 10) (MHT411 . 8) (LGV304 . 15))
```

```
CL-USER> *notes-etudiant2*
((INF353 . 14) (INF355 . 12) (INF351 . 12) (INF157 . 10) (MHT411 . 8))
```

2. Écrire une fonction `moyenne-etudiant` (`notes-etudiant`) qui calcule la moyenne d'un étudiant à partir de cette liste. On ne considérera pour calculer la moyenne que les notes présentes dans la liste de l'étudiant.

```
CL-USER> (moyenne-etudiant *notes-etudiant1*)
12
CL-USER> (moyenne-etudiant *notes-etudiant2*)
56/5
```

3. Écrire une fonction `note-si-present` (ue `notes-etudiant`) qui retourne la note obtenue à l'UE, si l'étudiant était présent à l'UE, NIL sinon. Exemples :

```
CL-USER> (note-si-present 'LGV304 *notes-etudiant1*)
15
CL-USER> (note-si-present 'LGV304 *notes-etudiant2*)
NIL
```

On mémorise les résultats de tous les étudiants dans une table de hachage. La *clé* est le nom de l'étudiant (une chaîne de caractères) et la *valeur* la liste d'association correspondant aux notes de l'étudiant.

4. Écrire l'expression qui permet de créer une telle table de hachage et de la mémoriser dans une variable `*notes-etudiants-s5*`.
5. En supposant que l'étudiant dont les notes sont mémorisées dans la variable `*notes-etudiant1*` se nomme "Pacman", écrire l'expression qui permet de rentrer ses notes dans la table de hachage.
6. Écrire une fonction `moyenne-ue` (ue `table`) qui calcule la moyenne d'une UE à partir des notes des étudiants contenues dans la table de hachage. Exemple (supposant que l'on a aussi rentré pour un deuxième étudiant, les notes mémorisées dans la variable `*notes-etudiant2*`) :

```
CL-USER> (moyenne-ue 'INF353 *notes-etudiants-s5*)
29/2
CL-USER> (moyenne-ue 'LGV304 *notes-etudiants-s5*)
15
```

Exercice 4 (3pts)

Soit le programme suivant :

```
(defclass a () ())

(defclass a1 (a) ())
(defclass a2 (a) ())
(defclass a21 (a2) ())
(defclass a22 (a2) ())

(defclass b () ())
(defclass b1 (b) ())
(defclass b11 (b1) ())
(defclass b12 (b1) ())
(defclass c () ())
(defclass b2 (b c) ())

(defgeneric m (o1 o2)
  (:documentation "une méthode qui s'applique à plusieurs objets"))

(defmethod m ((o1 a2) (o2 b12)) "a2 b12")
(defmethod m ((o1 a21) (o2 b1)) "a21 b1")
(defmethod m ((o1 a22) (o2 b11)) "a22 b11")
(defmethod m ((o1 a1) (o2 b1)) "a1 b1")

(defparameter *a21* (make-instance 'a21))
(defparameter *a22* (make-instance 'a22))
(defparameter *b11* (make-instance 'b11))
(defparameter *b12* (make-instance 'b12))
```

1. Dessiner la hiérarchie des classes définie par ce programme.
2. Quel est le résultat des deux appels suivants?
 - (a) (m *a22* *b11*)
 - (b) (m *a21* *b12*)

T.S.V.P

Exercice 5 (4pts)

Écrire une macro `with-array` (`array environnement &body body`) où

- `array` est une expression retournant un tableau,
- `environnement` est une liste de listes contenant chacune un nom de variable lexicale suivi d'une suite d'indices correspondant à une case du tableau et
- `body` est une suite d'expressions.

Cette macro crée un contexte de variables lexicales (avec un `let`) ayant chacune pour valeur initiale la valeur de la case du tableau correspondant à la suite d'indices, puis elle évalue les expressions du `body` dans ce contexte. Exemples :

```
CL-USER> (with-array (make-array 4 :initial-contents '(1 2 3 4)) ((x 0) (y 3))
           (list x y))
```

```
(1 4)
```

```
CL-USER> (defparameter *tab*
           (make-array '(2 3) :initial-contents '((1 2 3) (4 5 6))))
```

```
*TAB*
```

```
CL-USER> (with-array *tab* ((x 0 0) (y 1 2) (z 0 2))
           (format t "with-array~%")
           (list x y z))
```

```
with-array
```

```
(1 6 3)
```

FIN