DEVUIP Service Scolarité	ANNÉE UNIVERSITAIRE 2012/2013 2IÈME SESSION DE PRINTEMPS **Parcours :** IN501 **Code UE :** IN5011 **Épreuve :** Programmation 3 **Date :** juin 2013 **Heure :** **Durée :** 1h30 Documents : autorisés Épreuve de Mme Irène Durand	

Le barème est donné à titre **indicatif**. Le sujet comporte 3 pages.

Exercice 1 *(4pts)*

Les notes d'un examen sont comprises entre 0 et n (bornes comprises). On dispose d'un tableau histogramme h de taille $n+1$ contenant dans chaque case $h[i]$ le nombre de copies d'examen avec la note i. Écrire une fonction `moyenne-histo (h)` qui retourne la moyenne des notes. Exemples :

```
CL-USER> (moyenne-histo #(0 1 3 2 1 0)) ;; 7 notes entre 0 et 5
17/7
CL-USER> (coerce (moyenne-histo #(0 1 3 2 1 0)) 'float)
2.4285715
```

Exercice 2 *(3pts)*

Écrire une fonction récursive `fpower (f n)` qui prend en paramètre une fonction unaire `f` et une entier `n` et qui retourne la fonction unaire f^n. Noter que f^0 est la fonction identité et $f^1 = f$. Exemples :

```
CL-USER> (fpower #'1+ 5)
#<CLOSURE (LAMBDA (X) :IN FPOWER) {10041F4FEB}>
CL-USER> (funcall (fpower #'1+ 5) 4)
9
CL-USER> (funcall (fpower #'1+ 0) 4)
4
CL-USER> (funcall (fpower #'1+ 1) 4)
5
```

Exercice 3 *(6pts)*

Soit la macro `swap` suivante :

```
(defmacro swap (x y)
  `(let ((z ,x))
     (setf ,x ,y
           ,y z)
     (list ,x ,y)))
```

1. Compléter le scénario suivant :

```
CL-USER> (macroexpand-1 '(swap *x* *y*)) ;; Réponse A
```

```
CL-USER> (macroexpand-1 '(swap (aref *t* 1) (aref *t* 3))) ;; Réponse B
```

```
CL-USER> (setf *t* #(0 1 2 3)) ;; Réponse C

CL-USER> (swap (aref *t* 1) (aref *t* 3)) ;; Réponse D

CL-USER> *t* ;; Réponse E
```

2. Expliquer le comportement suivant :

```
CL-USER> (let ((x 1)
               (z 3))
           (swap x z))
; in: LET ((X 1) (Z 3))
;     (LET ((X 1) (Z 3))
;       (SWAP X Z))
;
; caught STYLE-WARNING:
;   The variable Z is defined but never used.
;
; compilation unit finished
;   caught 1 STYLE-WARNING condition
(1 1)
```

3. Modifier `swap` de manière à obtenir le comportement suivant :

```
CL-USER> (let ((x 1)
               (z 3))
           (swap x z))
(3 1)
```

Exercice 4 *(7pts)*
On se place dans le cadre d'une application gérant des comptes bancaires.

```
(defclass compte () ...)
(defgeneric transfert (montant compte1 compte2)
  :documentation "tranfere MONTANT du COMPTE1 vers COMPTE2")
```

1. Définir la classe `compte` avec un créneau `solde`, un accesseur `solde`, un mot-clé d'initialisation `:solde` et une valeur initiale de 0.00.

```
CL-USER> (defparameter *compte1* (make-instance 'compte :solde 10000))
*COMPTE*
```

2. Modifier l'implémentation de l'opération `print-object` de sorte que le solde soit affiché comme indiqué dans l'exemple qui suit.

```
CL-USER> *compte1*
#<COMPTE {AFC7761}> solde: 10000.00
```

3. Implémenter l'opération `transfert` sans vérification sur le solde des comptes.

```
CL-USER> (defparameter *compte2* (make-instance 'compte :solde 500))
*COMPTE2*
CL-USER> *compte2*
#<COMPTE {B01AAD9}> solde: 500.00
CL-USER> (transfert 300 *compte1* *compte2*)
800
CL-USER> *compte1*
#<COMPTE {AF7BEA1}> solde: 9700.00
CL-USER> *compte2*
#<COMPTE {B01AAD9}> solde: 800.00
```

   Certains comptes sont *plafonnés*. Le fait d'être plafonné est capturé par une classe *mixin* `plafonne-mixin` avec un créneau `plafond` destiné à mémoriser la valeur du plafond.

```
(defclass plafonne-mixin ()
  ((plafond :initarg :plafond :reader plafond :initform nil)))
```

4. En utilisant la classe mixin `plafonne-mixin`, définir une classe concrète `compte-plafonne`.
5. Modifier l'implémentation de `print-object` de manière à obtenir l'affichage suivant

```
CL-USER> (defparameter *cp*
           (make-instance 'compte-plafonne :solde 1000 :plafond 5000))
*CP*
CL-USER> *cp*
#<COMPTE-PLAFONNE {B07EAD9}> plafond: 5000.00 solde: 1000.00
```

6. En utilisant une méthode `:before` sur l'opération `transfert` ainsi que la macro `assert`, faire en sorte qu'une erreur soit signalée lorsque le plafond est dépassé.

```
CL-USER> (transfert 4500 *compte1* *cp*)

The assertion (< (+ (SOLDE COMPTE2) MONTANT) (PLAFOND COMPTE2)) failed.
   [Condition of type SIMPLE-ERROR]
...
CL-USER> (transfert 500 *compte1* *cp*)
1500
```

FIN