Ministère de l'enseignement supérieur et de la recherche scientifique
Université Djillali BOUNAAMA - Khemis Miliana (UDBKM)
Faculté des Sciences et de la Technologie
Département de Mathématiques et d'Informatique

Chapitre 4

Programmation Fonctionnelle
Exercices Corrigés

MI-GLSD-M1-UEM213 : Paradigmes de langages de Programmation

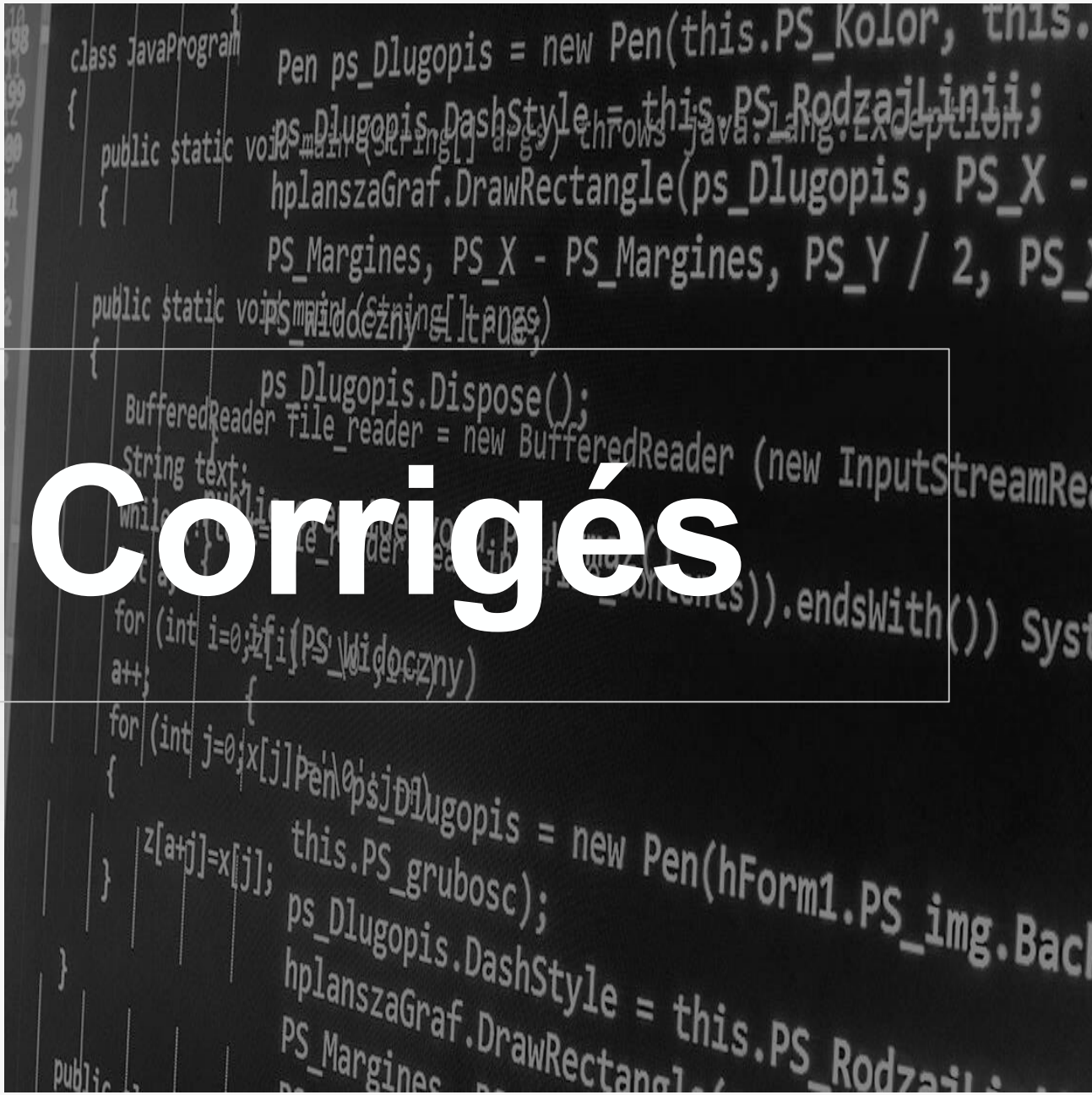
Noureddine AZZOUZA

n.azzouza@univ-dbkm.dz

Plan du Cours

1. Exercices Corrigés
2. Exercices d’examens

Exercices Corrigés



Exercices 01

- ✓ Doubler les éléments d'une liste
- ✓ **Solution 01:**

```
(DE Double(L)
   (IF (NULL L)
     L
     (Cons (Car L) (Cons (Car L)(Double (Cdr L)) ) )
   )
)
```

Exercices 01

- ✓ Doubler les éléments d'une liste
- ✓ **Solution 02:**

```
(DE Double (L)
   (IF (NULL L) L
     (IF (Consp (Car L))
        (Cons (Double (Car L)) (Double (Cdr L)))
        (Cons (Car L) (Cons (Car L)(Double (Cdr L)) ) )
   )
)
```

Exercices 02

- ✓ Suite de **Fibonacci**
- ✓ **Solution :**

```
(DE Fibonacci (N)
   (IF (EquaL N 1) 1
     (+(Fibonacci (- N 1)) (Fibonacci (- N 2))
   )
)
```

Exercices 03

- ✓ Calculer la **longueur** d'une liste. (Nombre d'éléments)
- ✓ **Solution :**

```
(DE Length (L)
   (IF (NULL L)
     0
     (+ 1 (Length (Cdr L)) )
   )
)
```

Exercices 04

- ✓ Calculer le nombre d'atoms d'une liste
- ✓ **Solution :**

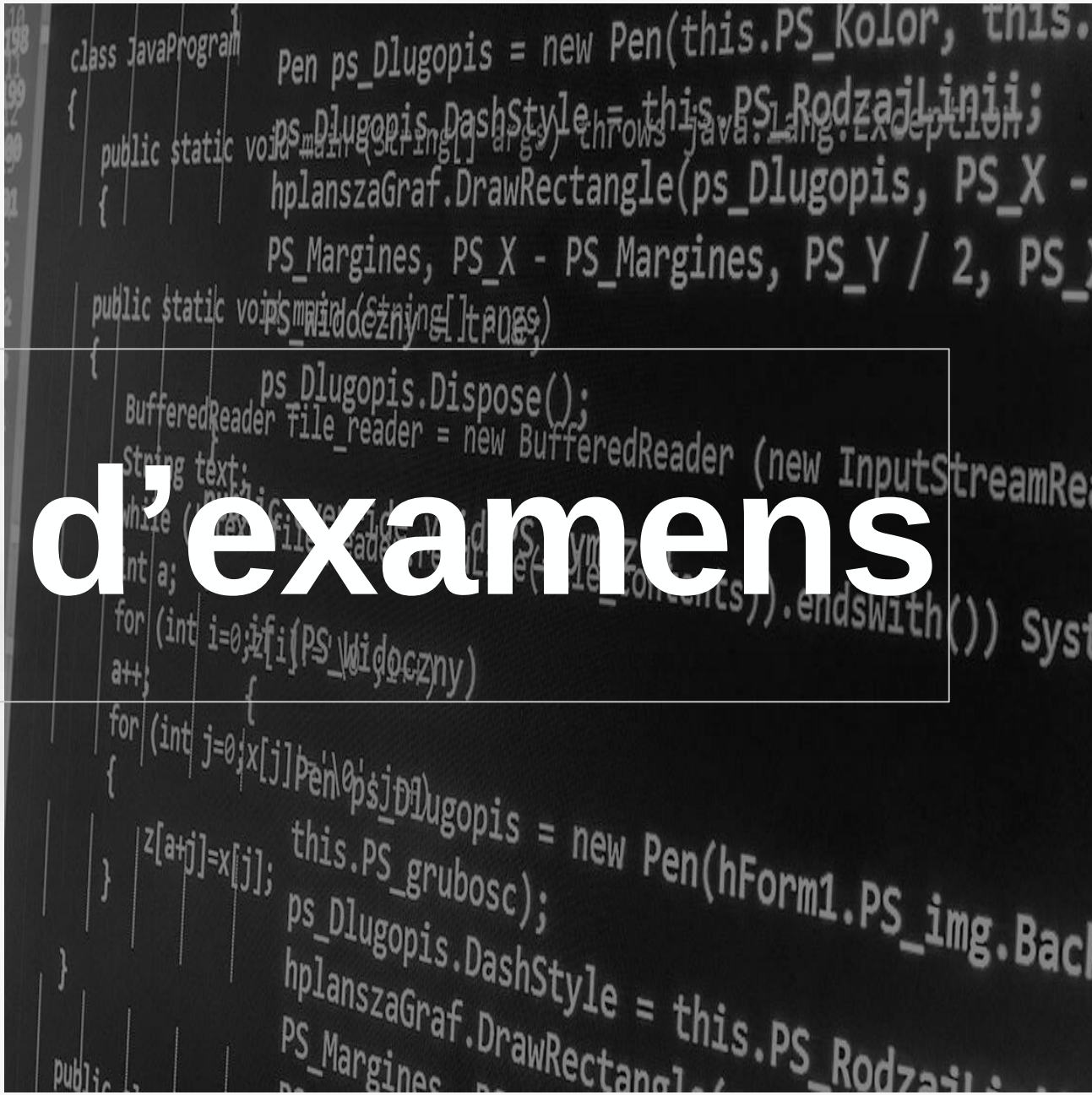
```
(DE NbAtoms(L)
   (IF (NULL L)  0
     (IF (ATOM L)  1
     (+ (NbAtoms(Car L))(NbAtoms(Cdr L)))
   )
)
```

Exercices 05

- ✓ Multiplier les éléments d'une liste
- ✓ **Solution :**

```
(DE Times(L)
   (IF (NULL (Cdr L))
     (Car L)
     (* (Car L) (Times (Cdr L)))
   )
)
```

Exercices d’examens



Exercices 01

Donner un programme en LISP définissant une fonction f ayant comme argument une liste L et retournant comme résultat une liste de deux sous-liste dont la première est formée par tous les éléments de rang **impair** dans L et la deuxième sous-liste est formée par tous les éléments de rang **pair** dans L.

Exemple : (f '(a b c d e)) = ((a c e) (b d))

```
(DE Impair(L)
   (IF (Consp L)
     (IF (Consp (Cdr L))
        (Cons (Car L)
              (Impair (Cdr (Cdr L))))
        L)
     Nil)
)
```

```
(DE f (L)
   (IF (Consp L)
     (Cons (Impair L)
           (Cons (Impair (Cdr L))
                 Nil )
           )
     Nil)
)
```

Exercices 02 : PLP-2014-2015-Ratt

Ecrire une fonction **Elem (i , L)** en Lisp purement fonctionnel, qui retourne une liste formée par les i^{ème} éléments (s'ils existent) de chaque liste contenue dans L. L étant une **liste de liste**.

Exemple : (Elem 3 '((a b c) (a b e f) (a c) (a d g) (a d h))) = (c e g h)

```
(DE Elem(i L)
   (IF (NULL L) Nil
     (IF (Pos i (Car L))
        (Cons (Pos i (Car L))
              (Elem (i (Cdr L)))
        (Elem (i (Cdr L)))
   )
)
```

```
(DE Pos (i S)
   (IF (= i 1)
     (Car S)
     (Pos (- i 1)(Cdr S))
   )
)
```

Exercices 03 : PLP-2015-2016-Examen

Ecrire un programme LISP, sans utiliser l'affectation ni la séquentielle, permettant **fusionner** deux listes **ordonnées** d'entiers.

Exemple : (Fusion '(1 3 5) '(2 2 4 5)) = (1 2 2 3 4 5 5).

```
(DE Fusion (x y)
   (IF (NULL x) y
     (IF (NULL y) x
          (IF(< (Car x) (Car y))
               (Cons (Car x) (Fusion (Cdr x) y))
               (Cons (Car y) (Fusion x (Cdr y)))
          )
     )
   )
)
```

Exercices 04 : PLP-2016-2017-Ratt

Donner un programme en Lisp fonctionnel, définissant une fonction retournant la liste de tous les nombres entiers inférieurs ou égal à un nombre positif donné N.

Exemple : (Tous 4) --> (0 1 2 3 4)

```
(DE Tous (N)
      (IF (= N 0)
         (CONS '0 '() )
         (CONS N (Tous (- N 1) ) )
      )
)
```

Exercices 05 : PLP-2015-2016-Examen

Donnez une fonction LISP, permettant d'éliminer toutes les valeurs en double dans une liste L.

```
(DE elim (L)
   (IF  (CONSP  L)
     (IF (appartient (CAR L) (CDR L))
        (elim  (CDR L))
        (CONS  (CAR L)  (elim  (CDR L)))
     )
   )
)
```

```
(DE  appartient  (a L)
    (IF (CONSP  L)
        (IF  (EQUAL  a  (CAR L))
             T
              (appartient  a  (CDR L))
        )
        NIL
    )
)
```

Exercices 06

On peut représenter un arbre de recherche binaire en Lisp par une liste à trois éléments : (ArbreG Racine ArbreD) où ArbreG et ArbreD sont eux même des listes. L'arbre vide NIL est représenté par la liste vide ( ). Ex : ((( ) 10 (( ) 20 ( ))) 30 (( ) 40 ( ))) représente l'arbre de racine 30, sont fils gauche est un arbre de racine 10 et son fils droit est un arbre de racine 40. 20 est le fils droit de 10 et ce dernier n'a pas de fils gauche (il est à nil). 40 et 20 sont des feuilles.

a) Définir en LISP les fonctions '**Valeur(Arb)**' , '**fg(Arb)**' et '**fd(Arb)**' qui retournent respectivement la valeur de la racine, le sous-arbre gauche et le sous-arbre droit de Arb.

b) Utiliser les fonctions ci-dessus pour écrire, en LISP, une fonction 'Niveau(v,A)' qui retourne comme résultat le niveau de v dans l'arbre A.

Exercices 06

```
(DE Valeur (p)
   (Car (Cdr p))
)
```

```
(DE fg (p)
   (Car p)
)
```

```
(DE fd (p)
   (Car (Cdr (Cdr p)))
)
```

```
(DE Niveau (v A)
   (IF (= v (Valeur A)) 0
     (IF (< v (Valeur A))
        (+ 1 (Niveau v (fg A)) )
        (+ 1 (Niveau v (fd A)) )
     ))
   )
)
```

Exercices 07 : PLP-2015-2016-Examen

Donner un programme en LISP, permettant de transformer un arbre binaire en une liste d'éléments en **inordre**. On représentera un arbre binaire non vide par une liste à trois éléments : (valeur_racine sous-arbre-gauche sous-arbre-droit).

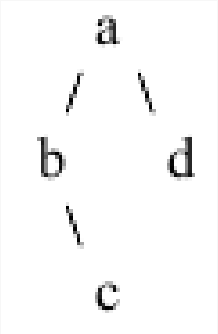
Un arbre vide sera représenté par une liste vide : ( )

Le parcours inordre d'un arbre binaire, consiste à visiter d'abord le sous-arbre gauche en inordre, puis la racine et enfin le sous-arbre droit en inordre.

Exemple, l'arbre binaire suivant :

a) sera représenté par : (a (b () (c () ())) (d () ())).

b) son parcours inordre génèrera la liste : (b c a d)



Exercices 07 : PLP-2015-2016-Examen


```
(DE fg (p)
   (Car (Cdr p))
)
```

```
(DE fd (p)
   (Car (Cdr (Cdr p)))
)
```

```
(DE concat (L1 L2)
   (IF (NULL L1)
       L2
       (CONS (CAR L1) (concat (CDR L1) L2) )
   )
)
```

```
(DE inordre (L)
  (IF (CONSP L)
    (IF (CONSP (fg L))
       (concat (inordre (fg L) (CONS (car L) (inordre (fd L)) ) )
       (CONS (car L) (inordre (fd L)) )
    )
  NIL
  )
)
```