

Programmation réursive

Annales corrigées
novembre 2001–septembre 2005

Anne Brygoo
Titou Durand
Maryse Pelletier
Christian Queinnec
Michèle Soria



Prochainement(?) :

UPMC/LI102 : compléments de cours

UPMC/LI102 : annales corrigées

UPMC/LI362 : annales corrigées

Missions de Paracamplus :

Paracamplus a pour vocation de procurer, à tous les étudiants en université, des ouvrages (annales corrigées, compléments de cours) et des environnements informatiques de travail, bref, des ressources pédagogiques propres à permettre, accélérer et entretenir l'apprentissage des connaissances enseignées.

Les ouvrages des éditions Paracamplus ont un format défini pour tenir commodément dans une poche et vous accompagner pendant vos déplacements en transports en commun. Ils bénéficient surtout d'une conception nous permettant de vous les proposer à des coûts très abordables.

Rafraîchir les annales tous les ans, procurer les compléments appropriés à chaque cours, vous proposer les meilleurs outils pour apprendre, tels sont les buts de ces ouvrages.

Découvrez-nous plus avant sur notre site **www.paracamplus.com**.

Il est interdit de reproduire intégralement ou partiellement la présente publication sans autorisation du Centre Français d'exploitation du droit de Copie (CFC) - 20 rue des Grands-Augustins - 75006 PARIS - Tél. : 01 44 07 47 70 /Fax : 01 46 34 67 19.

© 2005 Copyright retenu par les auteurs.

SARL Paracamplus

7, rue Viollet-le-Duc, 75009 Paris – France

ISBN 2-916466-00-2

Table des matières

1	Introduction	5
1.1	Bibliographie	6
2	Partiels	7
A	– Partiel – Novembre 2001	7
B	– Partiel – Novembre 2002	8
C	– Partiel – Novembre 2003	11
D	– Partiel – Novembre 2004	15
3	Examens	17
E	– Examen – Janvier 2002	17
F	– Examen – Septembre 2002	22
G	– Examen – Janvier 2003	26
H	– Examen – Septembre 2003	31
I	– Examen – Janvier 2004	35
J	– Examen – Janvier 2005	40
K	– Examen – Septembre 2005	45
4	Solutions	49
A	– Corrigé partiel – Novembre 2001	49
B	– Corrigé partiel – Novembre 2002	53
C	– Corrigé partiel – Novembre 2003	56
D	– Corrigé partiel – Novembre 2004	61
E	– Corrigé examen – Janvier 2002	65
F	– Corrigé examen – Septembre 2002	71
G	– Corrigé examen – Janvier 2003	78
H	– Corrigé examen – Septembre 2003	84
I	– Corrigé examen – Janvier 2004	87
J	– Corrigé examen – Janvier 2005	93
K	– Corrigé examen – Septembre 2005	99
5	Carte de référence	105
5.1	Spécification, signature, définition	105
5.2	Grammaire des types	105
5.3	Grammaire du langage	106

5.4	Commentaires	107
5.5	Bibliothèque	107
5.6	Suppléments	110
5.7	Bibliothèque graphique	111
5.8	Bibliothèque d'arbres	112

Chapitre 1

Introduction

Cet ouvrage est lié à l'enseignement « Programmation récursive » délivré au premier semestre de la première année de licence pour le parcours MIME (Mathématiques, Informatique, Mécanique, Électronique) de l'université Pierre et Marie Curie (UPMC). Le contenu de cet enseignement fait l'objet d'un ouvrage entier intitulé « Programmation récursive (en Scheme) » [BDP⁺04]. De nombreuses autres ressources pédagogiques sont également disponibles, en ligne, dans le site des enseignements d'informatique de licence en :

<http://www.licence.info.upmc.fr/>

Le présent ouvrage comporte une centaine de pages d'annales corrigées et commentées soit la majorité des « partiels » (ou devoirs sur table) et des examens (de première ou seconde session) de ces quatre dernières années. Travailler ces annales de concert avec le livre du cours est bien sûr fortement recommandé. Approfondir les textes étudiés pendant les travaux dirigés ou les travaux sur machine encadrés, s'entraîner avec un livre d'exercices tel que [MQRS99] sont également recommandés.

À la fin de ces annales se trouve un dernier chapitre décrivant le sous-ensemble du langage Scheme enseigné. Ce dernier chapitre, mis sous forme d'une page A4 pliée en trois et alors nommé la « carte de référence », est le seul document permis aux examens. Cet aide-mémoire (librement disponible sur le site mentionné plus haut) décrit les spécifications des quelque 80 fonctions prédéfinies nécessaires pour tous les exercices proposés (y compris la bibliothèque graphique et les bibliothèques appropriées pour les arbres binaires et généraux).

Une tradition de l'enseignement « Programmation récursive » est de diviser les examens en deux parties correspondant aux deux saisons rythmant l'enseignement. La première saison (7 semaines sur 12 en tout soit les 7 premiers chapitres de [BDP⁺04]) traite des récursions sur les entiers et les listes ; la seconde saison (les 5 semaines restantes traités dans les 5 derniers chapitres de [BDP⁺04]) aborde la récursion sur les arbres et culmine par le processus d'évaluation c'est-à-dire l'explication de l'évaluateur de Scheme en Scheme). La première partie des examens porte sur la première saison c'est-à-dire la récursion sur les entiers et sur les listes qui, si elle est totalement maîtrisée

permet d'obtenir entre 10 et 12 sur 20 à l'examen. Ce type de récursion est la connaissance fondamentale qui doit être acquise dans cet enseignement, les partiels traitent principalement de cette connaissance. Pour vous entraîner aux examens, commencez donc par vous entraîner avec les partiels!

Tous les partiels et examens présentés dans cet ouvrage sont corrigés. À de nombreux endroits plusieurs solutions sont présentées dans des styles de programmation variés. Ces solutions peuvent soit s'inspirer du schéma général de récursion [BDP⁺04, p. 89] soit utiliser des fonctionnelles [BDP⁺04, p. 117] permettant une écriture plus concise. Les corrigés des partiels sont plus riches en remarques pédagogiques.

Ne vous contentez pas de lire la solution ! Ce qui importe c'est de la concevoir, de la programmer, de la tester et enfin seulement de comparer votre réponse avec le corrigé. Ce n'est qu'au prix de votre implication personnelle que vous tirerez bénéfice de ces annales.

Sachez enfin que la durée des partiels est de 1 heure et demie mais que les examens s'étendent sur 2 heures. Vous pouvez soit vous placer dans les conditions de ces épreuves sur papier et sans machine, soit les travailler à votre rythme avec l'aide d'une machine. Cette dernière façon permet de mieux apprécier l'intérêt et l'originalité de certaines des examens. Les examens se dégustent mieux en dehors des examens.

Bonnes révisions !

1.1 Bibliographie

- [BDP⁺04] Anne Brygoo, Titou Durand, Maryse Pelletier, Christian Queinnec, et Michèle Soria. *Programmation récursive (en Scheme)*. Dunod, décembre 2004.
- [MQRS99] Luc Moreau, Christian Queinnec, Daniel Ribbens, et Manuel Ser-rano. *Recueil de petits problèmes en Scheme*. Scopos. Springer-Verlag, 1999.

Nous remercions tous nos collègues de l'enseignement « Programmation récursive » qui, pendant toutes ces années, ont lu, relu et amélioré ces épreuves.

Chapitre 2

Partiels

A – Partiel – Novembre 2001

▷ Exercice A.1

Question A.1.1 (2 points/30) Quelle est la valeur de `(cons 1 (cons 2 (list 5 4)))` ?

Question A.1.2 (2 points/30) Donnez une expression en Scheme ayant pour valeur `((1 2) (3))` en n'utilisant que des appels à `list` sans usage de la forme spéciale `quote`.

Question A.1.3 (3 points/30) Donnez une expression en Scheme ayant pour valeur `((1 2) (3))` en n'utilisant que des appels à `cons`. Il est possible de faire usage de la forme `quote`.

Question A.1.4 (3 points/30) Indiquer, en mentionnant leur nature (constructeur, reconnaisseur ou sélecteur), les fonctions que vous connaissez sur le type `LISTE` □.

▷ Exercice A.2

Un nombre naturel non nul n de représentation décimale $\overline{n_i n_{i-1} \dots n_1 n_0}$ (avec $n_i \neq 0$) est dit *théosophiquement équilibré* lorsque la somme des chiffres de rang pair qui le représentent en base décimale est égale à la somme des chiffres de rang impair ou, en termes plus formels, lorsque $n_0 + n_2 + \dots + n_{2j} = n_1 + n_3 + \dots + n_{2j+1}$ lorsque le nombre a un nombre pair $(2j + 2)$ de chiffres ou $n_0 + n_2 + \dots + n_{2j+2} = n_1 + n_3 + \dots + n_{2j+1}$ lorsque le nombre a un nombre impair $(2j + 3)$ de chiffres.

Aucun nombre à un seul chiffre n'est théosophiquement équilibré. Les multiples de 11 inférieurs à 100 sont les seuls nombres théosophiquement équilibrés entre 10 et 100, à savoir 11, 22, 33, ... 99. Parmi les nombres théosophiquement équilibrés à 3 chiffres, on trouve 121 ou 154 ou encore 352. Donnons simplement comme exemples de nombres théosophiquement équilibrés à 4 chiffres : 5445 ou 5742.

Question A.2.1 (4 points/30) Écrire une fonction `somme` prenant une liste d'entiers quelconques et calculant la somme de ces nombres. Ainsi :

`(somme (list 1 2 3 4))` → 10

Question A.2.2 (4 points/30) Écrire une fonction nommée `representation-liste` prenant un entier naturel non nul $n = \overline{n_i n_{i-1} \dots n_1 n_0}$ et retournant la liste des chiffres qui composent sa représentation décimale c'est-à-dire $(n_0 n_1 \dots n_i)$. Ainsi

`(representation-liste 245)` → (5 4 2)

Question A.2.3 (6 points/30) Écrire les fonctions nommées `pairs` (resp. `impairs`) prenant une liste et retournant la liste des termes de rang pair (resp. impair). On compte le rang à partir de zéro. Ainsi :

`(pairs (list 1 2 3 4 5))` → (1 3 5)

`(pairs (list 1 2 3 4 5 6))` → (1 3 5)

`(impairs (list "un" "deux" "trois"))` → ("deux")

`(impairs (list "un" "deux" "trois" "quatre"))`

→ ("deux" "quatre")

Question A.2.4 (6 points/30) Écrire un prédicat que l'on nommera `theosophiquement-equilibre ?` prenant un entier naturel non nul n et vérifiant que n est théosophiquement équilibré.

B – Partiel – Novembre 2002

▷ Exercice B.1

Question B.1.1 (2 points/20) Écrire la spécification et une définition de la fonction `factorielle` qui, étant donné un entier naturel n , retourne la factorielle de n .

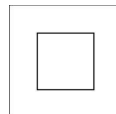
Question B.1.2 (2 points/20) Écrire la spécification et une définition de la fonction `produit-liste` qui, étant donnée une liste L de nombres retourne le produit des éléments de L . Par convention, le produit des éléments d'une liste vide est égal à 1.

▷ Exercice B.2

Cet exercice graphique traite de récursion sur les nombres. On dispose d'une fonction `dessin-carre` de spécification :

;;; `dessin-carre` : Nombre/compris entre 0 et 2/ -> Image

;;; (`dessin-carre c`) retourne le pourtour d'un carré de côté c centré en $(0,0)$



Par exemple : `(dessin-carre 1)` →

Noter que le dessin est toujours inscrit dans un cadre carré centré en $(0, 0)$ et de côté 2.

On considère la fonction `mystere` définie par :

```
(define (mystere n c)
  (if (= n 0)
      (image-vide)
      (overlay (dessin-carre c)
                (mystere (- n 1) (* c 0.5))))))
```

Question B.2.1 (1 point/20) Donner les applications successives de `mystere` (appels récursifs) qui sont effectuées lors de l'évaluation de `(mystere 3 2)`.

Question B.2.2 (1 point/20) Dessiner le résultat de l'évaluation de `(mystere 3 2)`.

Question B.2.3 (2 points/20) Donner la spécification de la fonction `mystere`.

▷ Exercice B.3

Cet exercice sur les listes est découpé en parties mais il forme un tout ; les titres des différentes parties font référence aux connaissances requises pour traiter les questions.

Exercices simples sur les listes

Question B.3.1 (2 points/20) Écrire la spécification et une définition de la fonction `au-moins-2?` qui, étant donnée une liste L d'éléments, retourne vrai si la liste L a au moins deux éléments et faux sinon. Attention à l'efficacité!

Question B.3.2 (2 points/20) Écrire la spécification et une définition de la fonction `ajout-1-au-car` qui, étant donnée une liste non vide L d'entiers naturels, retourne la liste obtenue en ajoutant 1 au premier élément de la liste L . Par exemple :

```
(ajout-1-au-car (list 2 5 1 4)) → (3 5 1 4)
```

Question B.3.3 (2 points/20) Écrire la spécification et une définition de la fonction `initialisation` qui, étant donnés un entier p et une liste L d'entiers naturels, retourne la liste formée de l'entier 1, de l'entier p et de tous les éléments de la liste L . Par exemple :

```
(initialisation 5 (list 4 3 7 9)) → (1 5 4 3 7 9)
```

Récursion sur les nombres

Voici une suite de listes :

- la liste 1 est la liste (1)
- la liste 2 est la liste (1 1)
- la liste 3 est la liste (2 1)
- la liste 4 est la liste (1 2 1 1)
- la liste 5 est la liste (1 1 1 2 2 1)
- la liste 6 est la liste (3 1 2 2 1 1)