

Numéro d'anonymat :

Examen - Décembre 2011

LI101

Durée 2 heures

Aucun document ni machine électronique n'est permis à l'exception de la carte de référence de Scheme.

Le sujet comporte 17 pages. Ne pas désagrafer les feuilles.

Répondre sur la feuille même, dans les boîtes appropriées. La taille des boîtes suggère le nombre de lignes de la réponse attendue. Le barème apparaissant dans chaque boîte n'est donné qu'à titre indicatif. Le barème total est lui aussi donné à titre indicatif : 70 points.

La clarté des réponses et la présentation des programmes seront appréciées. Les questions peuvent être résolues de façon indépendante. Il est possible, voire utile, pour répondre à une question, d'utiliser les fonctions qui sont l'objet des questions précédentes.

Pour vous faire gagner du temps, il ne vous est pas systématiquement demandé, en plus de la définition, la spécification entière d'une fonction. Bien lire ce qui est demandé : seulement la définition ? la signature et la définition ? la spécification et la définition ? seulement la spécification ?

Lorsque la signature d'une fonction vous est demandée, vous veillerez à bien préciser, avec la signature, les éventuelles hypothèses sur les valeurs des arguments.

Première partie

Exercices simples

Exercice 1

On considère les fonctions mutuellement récursives :

$$f(n) = \begin{cases} 1 & \text{si } n = 0 \\ f(n-1) + 2g(n-1) & \text{sinon} \end{cases} \quad \text{et} \quad g(n) = \begin{cases} 2 & \text{si } n = 0 \\ 2f(n-1) + g(n-1) & \text{sinon} \end{cases}$$

Question 1.1

Écrire la signature et une définition de la fonction Scheme **f**, ainsi que la signature et une définition de la fonction Scheme **g**, qui, étant donné un entier naturel n renvoie $f(n)$, respectivement $g(n)$.

Par exemple :

(f 1) → 5

(g 1) → 4

(f 2) → 13

(g 2) → 14

Numéro d'anonymat :

[3/70]

Question 1.2

Écrire la signature et une définition de la fonction Scheme `liste-fg` qui, étant donné un entier naturel n renvoie la liste des couples $(f(n), g(n)), (f(n-1), g(n-1)) \dots (f(0), g(0))$. Une attention particulière sera portée à l'efficacité de la fonction `liste-fg` (elle ne devra pas utiliser les fonctions `f` et `g`).

`(liste-fg 3) → ((41 40) (13 14) (5 4) (1 2))`

`(liste-fg 0) → ((1 2))`

Numéro d'anonymat :

[5/70]

Exercice 2

Dans une liste d'entiers naturels, certaines valeurs peuvent se répéter plusieurs fois consécutivement. On appelle *plage* une telle répétition d'une même valeur. Une plage est donc une sous-liste de la liste de départ contenant la succession de valeurs répétées.

Ainsi, par exemple, dans la liste (1 1 2 1 1 1 3), il y a 4 plages de valeurs : la plage (1 1), puis la plage (2), puis (1 1 1) et finalement la plage (3).

Question 2.1

Écrire la signature et une définition de la fonction Scheme **plage-elt** qui, étant donné un entier naturel x et une liste d'entiers naturels L renvoie la première plage de L si celle-ci est composée d'éléments égaux à x , ou renvoie la liste vide sinon.

Par exemple :

```
(plage-elt 1 '(1 1 2 1 1 1 3)) → (1 1)
```

```
(plage-elt 1 '(1 2 1 1 1 3)) → (1)
```

```
(plage-elt 1 '(2 1 1 1 3)) → ()
```

```
(plage-elt 1 '(11 38)) → ()
```

```
(plage-elt 5 '()) → ()
```

Numéro d'anonymat :

[3/70]

Question 2.2

Écrire une définition de la fonction Scheme **premiere-plage** qui, étant donné une liste d'entiers naturels L renvoie la première plage de cette liste, ou la liste vide si L est vide.

La signature de cette fonction est la suivante :

```
;;; premiere-plage: Liste[nat] -> Liste[nat]
```

Par exemple :

```
(premiere-plage '(1 1 2 1 1 1 3)) → (1 1)
```

```
(premiere-plage '(11 38)) → (11)
```

```
(premiere-plage '(42)) → (42)
```

```
(premiere-plage '()) → ()
```

[2/70]

Question 2.3

Écrire une définition de la fonction Scheme **decoupe-elt** qui, étant donné un entier naturel x et une liste d'entiers naturels L renvoie la liste obtenue en enlevant à L sa première plage si les éléments de cette première plage sont égaux à x , ou renvoie la liste L sinon.

La signature de cette fonction est la suivante :

```
;;; decoupe-elt : nat * LISTE[nat] -> LISTE[nat]
```

Numéro d'anonymat :

Par exemple :

```
(decoupe-elt 1 '(1 1 2 1 1 1 3)) → (2 1 1 1 3)
(decoupe-elt 2 '(1 1 2 1 1 1 3)) → (1 1 2 1 1 1 3)
(decoupe-elt 7 '(1 1 2 1 1 1 3)) → (1 1 2 1 1 1 3)
(decoupe-elt 11 '(11 11)) → ()
(decoupe-elt 5 '()) → ()
```

[3/70]

Question 2.4

Écrire une définition de la fonction Scheme `decoupe` qui, étant donné une liste d'entiers naturels L , renvoie la liste des éléments restant une fois les éléments de la première plage retirés de L . La fonction rendra la liste vide si L est vide.

La signature de cette fonction est la suivante :

```
;;; decoupe: Liste[nat] -> Liste[nat]
```

Par exemple :

```
(decoupe '(1 1 2 1 1 1 3)) → (2 1 1 1 3)
(decoupe '(11 38)) → (38)
(decoupe '(42 42 42)) → ()
(decoupe '()) → ()
```

[2/70]

Numéro d'anonymat :

Question 2.5

On souhaite maintenant obtenir la liste de toutes les plages présentes dans une liste L d'entiers naturels donnée.

Écrire la signature et une définition de la fonction Scheme `liste-plages` qui, étant donné une liste d'entiers naturels L renvoie la liste de toutes ses plages.

Par exemple :

```
(liste-plages '(1 1 2 1 1 1 3)) → ((1 1) (2) (1 1 1) (3))
```

```
(liste-plages '(11 38)) → ((11) (38))
```

```
(liste-plages '(42)) → ((42))
```

```
(liste-plages '()) → ()
```

[3/70]

Question 2.6

On considère donnée la fonction `length` suivante :

```
;;; length: Liste[alpha] -> nat
```

```
;;; (length L) rend le nombre d'éléments que contient L.
```

En utilisant une fonctionnelle, écrire la signature et une définition de la fonction Scheme `tailles-plages` qui, étant donné une liste de plages LP , telle celle obtenue par la fonction précédente, renvoie la liste des tailles, en nombre d'éléments, de chacune de ces plages.

Par exemple :

```
(tailles-plages '((1 1) (2) (1 1 1) (3))) → (2 1 3 1)
```

```
(tailles-plages '((11) (38))) → (1 1)
```

```
(tailles-plages (liste-plages '(42))) → (1)
```

```
(tailles-plages (liste-plages '())) → ()
```

Numéro d'anonymat :

[2/70]

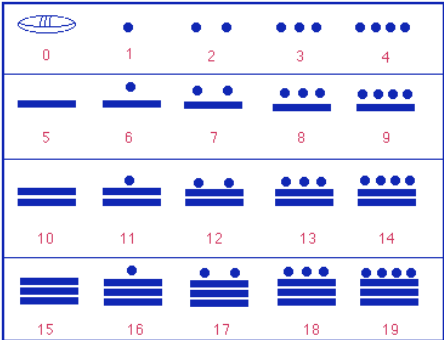




Deuxième partie

La numération maya

La numération maya est une numération de base 20 (*vigésimale*) qui utilise trois symboles pour représenter tous les nombres :

- le point qui désigne l'unité,
- la barre qui désigne la quantité cinq,
- l'œil qui désigne la quantité zéro.

Les chiffres mayas correspondent aux nombres de 0 à 19 de notre base. À partir de 20, les Mayas utilisent une numération de position, la lecture se faisant de haut en bas.

Tableau des chiffres mayas	Exemples de nombres mayas		
	27		$1 \cdot 20 + 7$
	358		$17 \cdot 20 + 18$
	340		$17 \cdot 20 + 0$
	112211		$14 \cdot 8000 + 0 \cdot 400 + 10 \cdot 20 + 11$

En Scheme nous représenterons les chiffres mayas par des couples et les nombres mayas par des listes de couples.

Numéro d'anonymat :

Chiffres mayas

Dans la numération maya un chiffre est représenté par des points et des barres ; en Scheme nous le représenterons par un couple dont le premier élément est égal au nombre de points et dont le second élément est égal au nombre de barres.

Par exemple le chiffre 17 (2 points et 3 barres) est représenté par le couple (2 3). Le chiffre 0 est représenté par le couple (0 0).

Remarquons que le nombre de points est toujours inférieur ou égal à 4 et que le nombre de barres est toujours inférieur ou égal à 3.

Nous appellerons **Glyphe** le type des couples représentant les chiffres mayas, c'est-à-dire **Glyphe** ::= COUPLE[Nat4 Nat3] où **Nat4** est le type des entiers de 0 à 4 et **Nat3** le type des entiers de 0 à 3. Nous appellerons *glyphe* un couple de type **Glyphe**.

Exercice 3

Question 3.1

Écrire la signature et une définition de la fonction **nb->glyphe** qui, étant donné un entier naturel n inférieur ou égal à 19, renvoie le glyphe correspondant à n . Par exemple :

(nb->glyphe 17) → (2 3) car $17 = 3 * 5 + 2$

(nb->glyphe 0) → (0 0) car $0 = 0 * 5 + 0$

(nb->glyphe 10) → (0 2) car $10 = 2 * 5 + 0$

[2/70]

Question 3.2

Écrire la signature et une définition de la fonction **glyphe->nb** qui, étant donné un glyphe G , renvoie le nombre correspondant à G . Par exemple :

(glyphe->nb '(1 3)) → 16

(glyphe->nb '(0 0)) → 0

(glyphe->nb '(4 0)) → 4

Numéro d'anonymat :

[2/70]

Nombres mayas

Dans la numération maya un nombre est représenté par une superposition de chiffres mayas. En Scheme nous représenterons cette superposition par une liste de glyphs :

- les glyphs correspondent aux valeurs obtenues par décomposition du nombre en base 20 ; par exemple, $112211 = 11 * 20^0 + 10 * 20^1 + 0 * 20^2 + 14 * 20^3$ et les glyphs correspondant aux valeurs 11, 10, 0 et 14 sont respectivement (1 2), (0 2), (0 0) et (4 2),
- la liste des glyphs est rangée en suivant l'ordre croissant des puissances de 20 ; 112211 est donc représenté par ((1 2) (0 2) (0 0) (4 2)).

Exercice 4

Question 4.1

Écrire la signature et une définition de la fonction `nb->maya` qui, étant donné un entier naturel n , renvoie la liste de glyphs correspondant à l'écriture de n dans la numération maya. Par exemple :

`(nb->maya 112211) → ((1 2) (0 2) (0 0) (4 2))`

`(nb->maya 340) → ((0 0) (2 3))`

`(nb->maya 17) → ((2 3))`

`(nb->maya 0) → ((0 0))`

[4/70]

Numéro d'anonymat :

Question 4.2

Écrire la signature et une définition de la fonction `maya->nb` qui, étant donnée une liste de glyphes, renvoie le nombre correspondant à cette liste. Par convention, cette fonction renvoie 0 si la liste est vide. Par exemple :

`(maya->nb '((1 2) (0 2) (0 0) (4 2)))` → 112211

`(maya->nb '((0 0) (2 3)))` → 340

`(maya->nb '((2 3)))` → 17

`(maya->nb '((0 0)))` → 0

`(maya->nb '())` → 0

Cette définition **ne doit pas** utiliser de **fonctionnelle**.

[4/70]

Question 4.3

Écrire une définition de la fonction `maya->nb-bis` de même spécification que la fonction `maya->nb`. Cette définition **doit** utiliser une **fonctionnelle**.

[5/70]

Numéro d'anonymat :

Troisième partie

Structures arborescentes

Arbres Nombres

Exercice 5

Un nombre entier naturel n peut se représenter à l'aide d'un arbre binaire A . Chaque feuille de A contient alors un chiffre de la représentation de n en base 10. Les nœuds internes d'un tel arbre contiennent, eux, la valeur -1 , qui n'est pas prise en compte dans la représentation de n .

Par exemple, les nombres 2015 et 42 sont représentés par les arbres donnés dans la Figure ??.

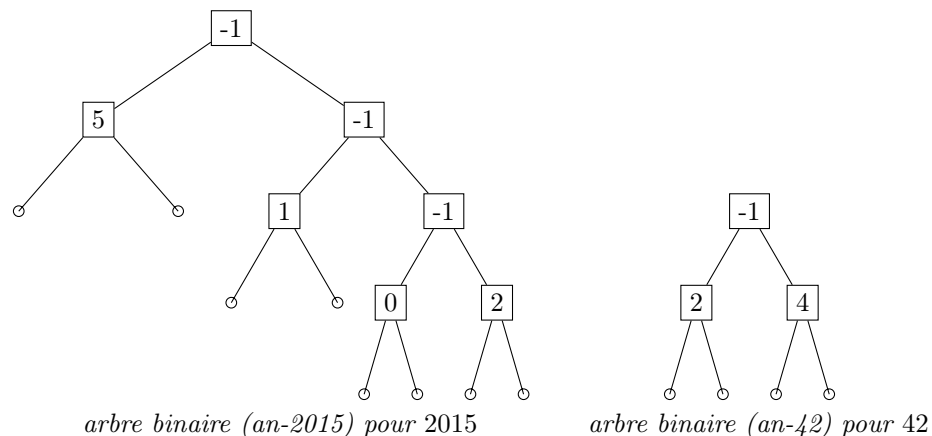


FIGURE 1 – Exemples de représentations de nombres par arbres binaires.

Le principe de la représentation d'un nombre dans un arbre A est le suivant. La valeur de l'arbre A est égale au nombre n que cet arbre représente. La valeur de A est calculée comme suit :

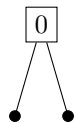
- si A est une feuille, sa valeur est donnée par son étiquette ;
- sinon, la valeur de A est obtenue en additionnant la valeur de son arbre gauche à 10 fois la valeur de son arbre droit.

Ainsi, par exemple, la valeur de l'arbre (an-2015) donné dans la Figure ?? est donc égale à la valeur de (ab-gauche (an-2015)) additionnée à 10 fois la valeur de (ab-droit (an-2015)).

Dans tout le reste de cet exercice, on appelle **ArbreNombre** un tel arbre.

Il faut remarquer qu'un arbre nombre est toujours un arbre binaire non vide, le plus petit arbre nombre est une feuille. Si un arbre nombre n'est pas une feuille alors ses deux sous-arbres sont des arbres nombres (donc non vides) et le sous-arbre gauche est une feuille.

Par exemple, l'entier 0 est représenté par l'arbre nombre suivant :



Dans tout cet exercice, on considère donnée la fonction suivante :

```
;;; ab-feuille: alpha -> ArbreBinaire[alpha]
;;; (ab-feuille e) rend l'arbre binaire dont l'étiquette est e
;;; et dont les sous-arbres gauche et droite sont vides.
```

Numéro d'anonymat :

Question 5.1

Écrire la signature et une définition de la fonction **an-2015** qui rend l'arbre représentant la valeur 2015 en base 10 donné dans la Figure ??.

[1.5/70]

Dans ce qui suit, on considère donnée la fonction **an-42** qui rend l'arbre nombre représentant l'entier 42 présenté dans la Figure ??.

Question 5.2

Écrire la signature et une définition de la fonction **ab-feuille?** qui, étant donné un arbre binaire, rend **#t** si cet arbre est une feuille.

[1.5/70]

Question 5.3

Pour construire un arbre A pour représenter un entier naturel n donné, il faut réaliser le travail inverse de celui qui est réalisé pour calculer la valeur de A . Si n est plus petit que 10, A est une feuille, sinon, A obtenu en construisant un nœud ayant la valeur -1 comme étiquette, et dont le sous-arbre gauche représente le chiffre des unités de n et dont le sous-arbre droit représente n privé de son chiffre d'unité.

Écrire la signature et une définition de la fonction **an-construit** qui, étant donné un entier naturel n rend l'arbre nombre qui le représente.

Par exemple, $(\text{an-construit } 2015)$ rend l'arbre représentant 2015 et $(\text{an-construit } 42)$ rend l'arbre représentant 42 donnés dans la Figure ?? .

Numéro d'anonymat :

[4/70]

Question 5.4

Écrire la signature et une définition de la fonction **an-unite** qui, étant donné un arbre nombre rend le chiffre des unités du nombre qu'il représente.

Par exemple : `(an-unite (an-2015))` \rightarrow 5

`(an-unite (an-42))` \rightarrow 2

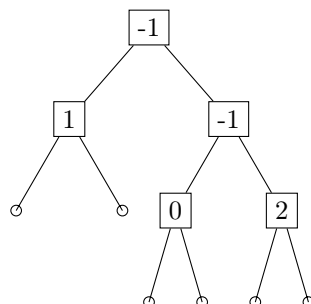
`(an-unite (an-construit 7))` \rightarrow 7

[2/70]

Question 5.5

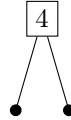
Écrire une définition de la fonction **an-quotient** qui, étant donné un arbre nombre **A** rend l'arbre nombre correspondant au quotient de la division de **A** par 10.

Par exemple : `(an-quotient (an-2015))` rend

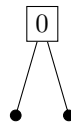


Numéro d'anonymat :

`(an-quotient (an-42))` rend



et `(an-quotient (an-construit 5))` rend



[3/70]

Question 5.6

Écrire la signature et une définition de la fonction **an-egal?** qui, étant donné deux arbres nombres **A** et **B** rend vrai si le nombre représenté par **A** est égal à celui représenté par **B**, et faux sinon.

Par exemple : `(an-egal? (an-2015) (an-2015))` → #t

`(an-egal? (an-2015) (an-42))` → #f

`(an-egal? (an-construit 42) (an-42))` → #t

`(an-egal? (an-42) (an-construit 420))` → #f

`(an-egal? (an-42) (an-construit 4))` → #f

Numéro d'anonymat :

[4/70]

Question 5.7

Écrire la signature et une définition de la fonction **an-chiffres** qui, étant donné un arbre nombre rend la liste des chiffres qui le composent, dans l'ordre décimal (le chiffre des unités en dernière position, le chiffre des dizaines en avant-dernière position, etc.).

Par exemple : `(an-chiffres (an-2015)) → (2 0 1 5)`

`(an-chiffres (an-42)) → (4 2)`

`(an-chiffres (an-construit 0)) → (0)`

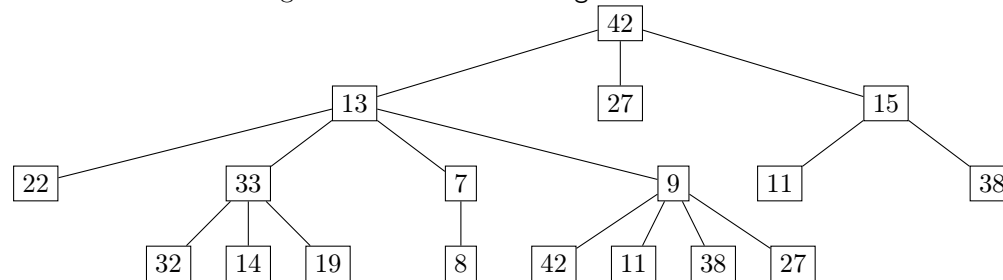
[4/70]

Numéro d'anonymat :

Arbres généraux

Exercice 6

On considère un arbre général comme l'arbre (**ag-ex**) suivant :



Question 6.1

Écrire la signature et une définition de la fonction **ag-valeurs-paires** qui, étant donné un arbre général **AG** contenant des entiers naturels rend la liste préfixe des nombres pairs présents dans cet arbre.

Par exemple : (**ag-valeurs-paires** (**ag-ex**)) → (42 22 32 14 8 42 38 38)

[4/70]

Question 6.2

On souhaite maintenant pouvoir récupérer la liste préfixe de toutes les valeurs de **AG** qui vérifient un prédicat donné.

Écrire la signature et une définition de la fonction **ag-valeurs** qui, étant donné un prédicat **pred?** ainsi qu'un arbre général **AG** rend la liste préfixe des valeurs présentes dans cet arbre qui vérifient le prédicat **pred?**.

Par exemple :

Numéro d'anonymat :

```
(ag-valeurs even? (ag-ex)) → (42 22 32 14 8 42 38 38)
(ag-valeurs odd? (ag-ex)) → (13 33 19 7 9 11 27 27 15 11)
(ag-valeurs negative? (ag-ex)) → ()
```

[4/70]

Question 6.3

En utilisant la fonction précédente, écrire une définition de la fonction **ag-valeurs-sup20** qui, étant donné un arbre général **AG** contenant des nombres rend la liste préfixe des valeurs présentes dans cet arbre qui sont strictement supérieures à 20.

Par exemple :

```
(ag-valeurs-sup20 (ag-ex)) → (42 22 33 32 42 38 27 27 38)
```

[2/70]