Numéro d'anonymat :

Examen - Janvier 2013
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 : 66 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

Suites arithmétiques

Exercice 1

Une suite arithmétique U est de la forme :

$$\begin{cases} U_0 = k \\ U_n = r + U_{n-1} \end{cases}$$

Par exemple, si on pose $k = 1$ et $r = 2$ alors la suite U est : $1, 3, 5, 7, 9, \ldots$ Le nombre k se nomme la *condition initiale* de la suite U et le coefficient additif r se nomme *la raison*.

Question 1.1

Donner une définition de la fonction `arith` dont la spécification est la suivante :

```
;;; arith: Nat * Nombre * Nombre -> Nombre
;;; (arith n k r) renvoie le terme d'indice n de la suite arithmétique
;;; de condition initiale k et de raison r
```

Par exemple :

`(arith 0 1 2)` → `1`

`(arith 1 1 2)` → `3`

`(arith 2 1 2)` → `5`

Numéro d'anonymat :

`(arith 3 1 2)` → `7`

`(arith 4 1 2)` → `9`

[2/66]

Question 1.2

Soit la fonction `mystere` suivante :

```
(define (mystere a b c)
  (if (= a 0)
      (list b)
      (let ((U (mystere (- a 1) b c)))
         (cons (+ c (car U)) U))))
```

Donner le résultat de l'évaluation de : `(mystere 2 1 2)`

En déduire la signature de la fonction `mystere`.

Expliquer en une phrase ce que réalise cette fonction selon vous.

[3/66]

Question 1.3

Donner une définition de la fonction `renverse` qui, à partir d'une liste `L`, renvoie la liste des éléments de `L` en ordre inverse. La signature de la fonction est la suivante :

```
;;; renverse : LISTE[alpha] -> LISTE[alpha]
```

Numéro d'anonymat :

Par exemple :

`(renverse (list))` → `()`

`(renverse (list #t))` → `(#t)`

`(renverse (list 1 2))` → `(2 1)`

`(renverse (list 4 2 1))` → `(1 2 4)`

`(renverse (list "un" "deux" "trois" "quatre"))` → `("quatre" "trois" "deux" "un")`

[4/66]

Question 1.4

Donner la signature et une définition de la fonction `liste-arith` telle que `(liste-arith n k r)` renvoie la liste $(U_0\ U_1\ \ldots\ U_n)$.

Par exemple :

`(liste-arith 0 1 2)` → `(1)`

`(liste-arith 1 1 2)` → `(1 3)`

`(liste-arith 2 1 2)` → `(1 3 5)`

`(liste-arith 3 1 2)` → `(1 3 5 7)`

`(liste-arith 4 1 2)` → `(1 3 5 7 9)`

Attention : vous n'aurez que la moitié des points si votre fonction utilise `arith`.

[3/66]

Numéro d'anonymat :

Deuxième partie

Opérations sur les polynômes

Le but de cet exercice est de définir un ensemble de fonctions capables de manipuler des polynômes.

Un polynôme à une variable est un objet mathématique que l'on définit comme étant la somme des termes $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$. Chaque terme du polynôme est de la forme a_kx^k dans lequel x est la variable du polynôme, k le *degré* du terme et a_k son coefficient. On appelle alors *degré du polynôme* le plus grand degré de ses termes.

Ainsi, un polynôme de degré n comme $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ est complètement défini par la liste des coefficients $(a_0\ a_1\ \ldots\ a_n)$. Par exemple, le polynôme $P_1(x) = 14 + x - 4x^2$, de degré 2 est déterminé par la liste de ses coefficients `(14 1 -4)`. De même, le polynôme $P_2(x) = x + 3x^2 + 5x^7$, de degré 7, est représenté par `(0 1 3 0 0 0 0 5)` et le polynôme constant $P_3(x) = 4$, de degré 0 est représenté par `(4)`. On s'intéresse, dans cet exercice, aux polynômes à coefficients entiers et on nous appelleront `Poly` leur type. C'est-à-dire que `Poly::=LISTE[int]`. Par ailleurs, on décide de représenter le polynôme nul par la liste `(0)`.

Remarque : Dans tout ce qui suit, nous pourrons supposer, sans le rajouter en hypothèse dans les spécifications, que tout polynôme contient au moins 1 coefficient, c'est à dire que la liste qui le représente est non vide.

Nous supposerons également que, mis à part pour le polynôme nul, le dernier coefficient de la liste est différent de 0.

Exercice 2

Dans ce qui suit, on supposera données les fonctions permettant de créer les polynômes P_1, P_2, P_3 ainsi que le polynôme nul :

```
;;; poly1: -> Poly
;;; (poly1) renvoie le polynôme P(x) = 14 + x - 4 x^2
(define (poly1)
  (list 14 1 -4))

;;; poly2: -> Poly
;;; (poly2) renvoie le polynôme P(x) = x + 3 x^2 + 5 x^7
(define (poly2)
  (list 0 1 3 0 0 0 0 5))

;;; poly3: -> Poly
;;; (poly3) renvoie le polynôme P(x) = 4
(define (poly3)
  (list 4))

;;; polynul: -> Poly
;;; (polynul) renvoie le polynôme nul P(x) = 0
(define (polynul)
  (list 0))
```

Numéro d'anonymat :

Question 2.1

Donner une définition du prédicat `p-nul?` qui teste si un polynôme est nul ou non. La signature de la fonction est la suivante :

```
;;; p-nul?: Poly -> bool
```

Par exemple :

`(p-nul? (poly2))` → `#f`

`(p-nul? (poly3))` → `#f`

`(p-nul? (polynul))` → `#t`

[2/66]

Question 2.2

Donner la signature et une définition de la fonction `p-degre` qui, étant donné un polynôme `P`, renvoie le degré de `P`.

`(p-degre (poly1))` → `2`

`(p-degre (poly2))` → `7`

`(p-degre (poly3))` → `0`

`(p-degre (polynul))` → `0`

[3/66]

Numéro d'anonymat :

Question 2.3

Donner la signature et une définition de la fonction `p-valeur` qui, étant donné un nombre `x` et un polynôme `P`, renvoie la valeur du polynôme `P` en `x`.

Indication. On rappelle le schéma de Hörner :

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = a_0 + x * (a_1 + x * (a_2 + x * (\ldots + x * a_n) \ldots))$$

Par exemple :

`(p-valeur 0.0 (poly1))` → `14.0`

`(p-valeur 1.0 (poly1))` → `11.0`

`(p-valeur  2.0 (poly1))` → `0.0`

`(p-valeur 2.5 (polynul))` → `0`

[4/66]

Question 2.4

On appelle *racine* d'un polyôme P une valeur x telle que $P(x) = 0$.

À l'aide de la fonction précédente, donner la signature et une définition du prédicat `p-racine?` qui, étant donné un nombre `x` et un polynôme `P`, teste si `x` est racine de `P`.

`(p-racine? 0.0 (poly1) )` → `#f`

`(p-racine? 1.0 (poly1) )` → `#f`

`(p-racine? 2.0 (poly1) )` → `#t`

`(p-racine? 2.5 (polynul) )` → `#t`

[1/66]

Numéro d'anonymat :

Exercice 3

On s'intéresse dans cet exercice à l'addition entre polynômes.

On rappelle que l'addition de deux polyômes P et P' est donné par le polynôme P'' dont chacun des coefficients correspond à l'addition des coefficents des termes de même degré dans P et P'.

Attention, les degrés des deux polynômes ne sont pas forcément égaux. Si $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ et $P'(x) = a'_0 + a'_1x + a'_2x^2 + \ldots + a'_px^p$ avec $p > n$, alors l'addition de P et P' donne le polynôme $P''(x) = (a_0 + a'_0) + (a_1 + a'_1)x + (a_2 + a'_2)x^2 + \ldots + (a_n + a'_n)x^n + a'_{n+1}x^{n+1} + \ldots + a'_px^p$

Ainsi l'addition de P_1 et P_2 de l'exercice précédent donne le polynôme P suivant :

	P_1	14	1	−4					
+	P_2	0	1	3	0	0	0	0	5
	P	14	2	−1	0	0	0	0	5

Question 3.1

Donner une définition de la fonction `p-add` qui effectue l'addition de deux polynômes. La signature de la fonction est la suivante :

```
;;;p-add: Poly * Poly -> Poly
```

Par exemple :

`(p-add (poly1) (polynul))` → `(14 1 -4)`

`(p-add (polynul) (poly2))` → `(0 1 3 0 0 0 0 5)`

`(p-add (poly1) (poly2))` → `(14 2 -1 0 0 0 0 5)`

[4/66]

Numéro d'anonymat :

Question 3.2

À l'aide de la fonction précédente, donner la signature et une définition **récursive** de la fonction `p-add-liste` qui, étant donné une liste de polynômes `L=(P1 P2 ... Pn)` revoie le polynôme correspondant à l'addition de tous les polynômes `P1 + P2 + ... + Pn`.

Attention : cette fonction ne **doit pas** utiliser de fonctionnelle.

Par exemple :

`(p-add-liste (list))` → `(0)`

`(p-add-liste (list (poly1) (poly2)))` → `(14 2 -1 0 0 0 0 5)`

`(p-add-liste (list (polynul) (poly1) (poly2) (poly3)))` → `(18 2 -1 0 0 0 0 5)`

[3/66]

Question 3.3

Donner une définition de la fonction `p-add-liste-fct` de même spécification que `p-add-liste` en utilisant une **fonctionnelle**.

Par exemple :

`(p-add-liste-fct (list))` → `(0)`

`(p-add-liste-fct (list (poly1) (poly2)))` → `(14 2 -1 0 0 0 0 5)`

`(p-add-liste-fct (list (polynul) (poly1) (poly2) (poly3)))` → `(18 2 -1 0 0 0 0 5)`

[2/66]

Numéro d'anonymat :

Exercice 4

La multiplication de deux polynômes P et P' est un peu plus complexe à mettre en œuvre. L'idée générale est d'appliquer le même principe que lors de la multiplication de deux nombres : il faut faire la somme de tous les produits partiels de P' avec chacun des termes de P.

Ainsi, si $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$, alors la multiplication de P et P' donne le polynôme $P''(x) = a_0 * P'(x) + a_1 * x * P'(x) + a_2 * x^2 * P'(x) + \ldots + a_n * x^n * P'(x)$.

Ainsi la multiplication de P_1 et P_2 de l'exercice précédent donne le polynôme P suivant :

	P_1	14	1	-4							
$\times$	P_2	0	1	3	0	0	0	0	5		
	$14 * P_2$	0	14	42	0	0	0	0	70		
$+$	$1 * x * P_2$		0	1	3	0	0	0	0	5	
$+$	$-4 * x^2 * P_2$			0	-4	-12	0	0	0	0	-20
	P	0	14	43	-1	-12	0	0	70	5	-20

Nous avons donc besoin de définir des fonctions permettant la multiplication d'un polynôme par une constante ainsi que la multiplication d'un polynôme par la variable x.

Question 4.1

Donner la signature et une définition de la fonction `p-mult-cst` qui renvoie le polynôme résultant de la multiplication d'un polynôme par une constante. Vous pouvez, ou non, utiliser une fonctionnelle. La signature de la fonction est la suivante :

```
;;; p-mult-cst: int * Poly -> Poly
```

Par exemple :

`(p-mult-cst 2 (poly1))` → `(28 2 -8)`

`(p-mult-cst 2 (poly2))` → `(0 2 6 0 0 0 0 10)`

`(p-mult-cst 2 (poly3))` → `(8)`

`(p-mult-cst 2 (polynul))` → `(0)`

[2/66]

Numéro d'anonymat :

Question 4.2

Donner une définition de la fonction `p-mult-x` qui, étant donné un polynôme `P` revoie le polynôme `x*P`. La signature de la fonction est la suivante :

```
;;; p-mult-x : Poly -> Poly
```

Remarquons que, si `P` n'est pas le polynôme nul, la multiplication par la variable `x` correspond au simple décalage des coefficients de `P` dans la liste. Si `P` est le polynôme nul, alors la multiplication de `P` par une variable donne à nouveau le polynôme nul.

`(p-mult-x (poly1))` → `(0 14 1 -4)`

`(p-mult-x (poly2))` → `(0 0 1 3 0 0 0 0 5)`

`(p-mult-x (poly3))` → `(0 4)`

`(p-mult-x (polynul))` → `(0)`

[2/66]

Question 4.3

Nous allons maintenant effectuer la multiplication de deux polynômes.

Soit P_1 et P_2 deux polynômes. On peut remarquer que si P_1 est de degré supérieur ou égal à 1, il s'écrit sous la forme $P_1(x) = a_0 + x * P'_1(x)$ (par exemple, le polynôme $P(x) = 1 + 2x + 3x^2 + 4x^3$ s'écrit aussi sous la forme $P(x) = 1 + x * P'(x)$ avec $P'(x) = 2 + 3x + 4x^2$). La multiplication de P_1 par P_2 est donc donnée par $P_1(x) * P_2(x) = a_0 * P_2(x) + x * (P'_1(x) * P_2(x))$.

À l'aide de cette remarque, donner la une définition de la fonction `p-mult` qui effectue la multiplication de deux polynômes. La signature de la fonction est la suivante :

```
;;; p-mult: Poly * Poly -> poly
```

Par exemple :

`(p-mult (poly1) (poly2))` → `(0 14 43 -1 -12 0 0 70 5 -20)`

`(p-mult (poly1) (poly3))` → `(56 4 -16)`

`(p-mult (poly3) (polynul))` → `(0)`

`(p-mult (poly2) (poly3))` → `(0 4 12 0 0 0 0 20)`

Numéro d'anonymat :

[5/66]

Troisième partie

Arbres binaires de recherche et ensembles

Notre objectif est d'illustrer l'utilisation des arbres binaires de recherche pour implémenter une structure d'ensemble mathématique.

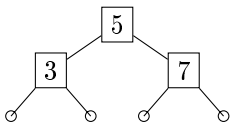
Remarque : pour simplifier les exercices, on supposera que les ensembles ne contiennent que des nombres.

Exercice 5

La barrière d'abstraction des ensembles est construite sur la barrière d'abstraction des arbres binaires ainsi que les fonctions de manipulation des arbres binaires de recherche étudiées en cours.

On appellera `Ensemble` le type des ensembles de nombres tel que `Ensemble ::= ArbreBinRech`

Par exemple, l'ensemble $E = \{3, 5, 7\}$ pourra être représenté par l'arbre binaire de recherche suivant :



Numéro d'anonymat :

Question 5.1

L'ensemble vide, noté $\emptyset$ en mathématiques, sera représenté par un arbre binaire vide. En déduire une définition de la fonction `vide` dont la spécification est la suivante :

```
;;; vide: -> Ensemble
;;; (vide) retourne l'ensemble vide
```

[1/66]

Question 5.2

Un *singleton* est un ensemble contenant un unique élément, comme par exemple $\{2\}$ (ensemble contenant le nombre 2) ou $\{-1\}$ (ensemble contenant le nombre -1). Un singleton $\{n\}$ où n est un nombre quelconque sera représenté par un arbre binaire dont l'unique nœud a pour étiquette est n.

Donner la signature et une définition de la fonction `singleton` permettant de construire un ensemble singleton $\{n\}$ à partir d'un nombre n.

[1.5/66]

Question 5.3

Donner la signature et une définition du prédicat `vide?` permettant, à partir d'un ensemble `E`, de vérifier s'il s'agit d'un ensemble vide ou non. On aura par exemple :

`(vide? (vide))` $\rightarrow$ `#t`

`(vide? (singleton 2))` $\rightarrow$ `#f`

[1.5/66]

Numéro d'anonymat :

Question 5.4

On rappelle la signature de la fonction de parcours d'un arbre binaire en ordre infixe :

```
;;; liste-infixe: ArbreBin[Alpha] -> LISTE[Alpha]
```

Par exemple, on aura :

```
(liste-infixe (ab-noeud 5
                        (ab-noeud 3 (ab-vide) (ab-vide))
                        (ab-noeud 7 (ab-vide) (ab-vide))))
```

→`(3 5 7)`

Rappel : la liste infixe d'un arbre binaire de recherche retourne les étiquettes triées en ordre croissant.

Donner une définition de la fontion `liste-infixe`.

[4/66]

A partir de la fonction `liste-infixe` on peut déduire une fonction permettant d'afficher les ensembles :

```
;;; ens : Ensemble -> LISTE[int]
;;; (ens E)  retourne la liste (triée) des éléments de E
(define (ens E)
  (liste-infixe E))
```

Exercice 6

On rappelle ci-dessous la fonction d'ajout dans un arbre binaire de recherche :

```
;;; abr-ajout : Nombre * ArbreBinRech -> ArbreBinRech
;;; (abr-ajout x ABR) rend l'arbre binaire de recherche obtenu en rajoutant
;;; x dans ABR. Si x est déjà dans ABR, celui-ci n'est pas modifié.
```

Numéro d'anonymat :

```
(define (abr-ajout x ABR)
  (if (ab-noeud? ABR)
      (let ((e (ab-etiquette ABR)))
        (cond ((= x e) ABR)
              ((< x e) (ab-noeud e (abr-ajout x (ab-gauche ABR)) (ab-droit ABR)))
              (else (ab-noeud e (ab-gauche ABR)  (abr-ajout x (ab-droit ABR))))))
      ;; cas vide
      (ab-noeud x (ab-vide) (ab-vide))))
```

Question 6.1

Dessiner l'arbre binaire de recherche correspondant à l'expression :

```
(abr-ajout 4 (abr-ajout 9 (abr-ajout 2 (abr-ajout 9 (abr-ajout 7 (ab-vide))))))
```

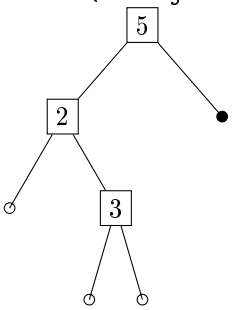
[2/66]

Question 6.2

Le principe de la fusion entre deux arbres binaires de recherche `ABR1` et `ABR2` est le suivant :
- si le second arbre `ABR2` est vide alors la fusion de `ABR1` et `ABR2` est simplement `ABR1`.
- si le second arbre `ABR2` est non-vide, alors la fusion de `ABR1` et `ABR2` correspond à la fusion de deux arbres `ABR1+` et `ABR2-` tels que :
  - l'arbre `ABR1+` est l'arbre `ABR1` auquel on a ajouté l'étiquette de `ABR2`
  - l'arbre `ABR2-` est la fusion des deux sous-arbres gauche et droit de `ABR2`

On aura ainsi, par exemple :

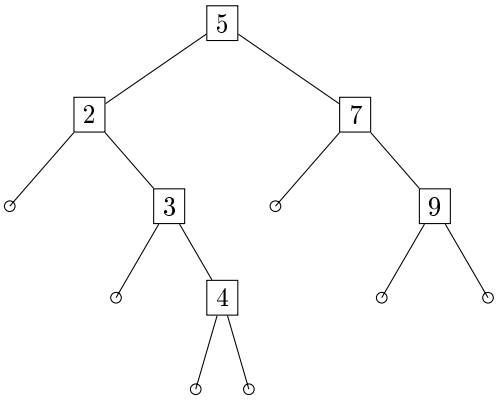
`(abr-fusion (abr-ajout 3 (abr-ajout 2 (abr-ajout 5 (ab-vide)))) (ab-vide))` →




```
(abr-fusion (abr-ajout 3 (abr-ajout 2 (abr-ajout 5 (ab-vide))))
            (abr-ajout 4 (abr-ajout 9 (abr-ajout 2 (abr-ajout 7 (ab-vide))))))
```

Numéro d'anonymat :

→



Donner une définition de la fonction `abr-fusion` qui, à partir de deux arbres binaires de recherche, en retourne la fusion selon les principes décrits ci-dessus. La spécification de cette fonction est la suivante :

```
;;; abr-fusion: ArbreBinRech * ArbreBinRech -> ArbreBinRech
```

[5/66]

Question 6.3

L'opération d'union entre deux ensembles permet de construire des ensembles quelconques.

Par exemple, l'ensemble $\{1, 7\}$ peut se construire comme l'union des singtelons $\{1\}$ et $\{7\}$ que l'on note $\{1\} \cup \{7\}$. Ensuite on peut aussi construire $\{1, 4, 7, 9\} = \{1, 7\} \cup \{4, 9\}$ avec $\{4, 9\} = \{4\} \cup \{9\}$, etc.

En scheme, on considérera une fonction `union` telle que, par exemple :

`(ens (union (singleton 1) (vide)))` → `(1)`

`(ens (union (vide) (singleton 1) ))` → `(1)`

`(ens (union (singleton 1) (singleton 7)))` → `(1 7)`

`(ens (union (singleton 9) (singleton 4)))` → `(4 9)`

```
(ens  (union (union (singleton 1) (singleton 7))
                    (union (singleton 4) (singleton 9))))
```

→`(1 4 7 9)`

Numéro d'anonymat :

En s'inspirant des fonctions précédentes, donner la signature et une définition de la fonction `union`.

[2/66]

Question 6.4

En exploitant la structure des arbres binaires de recherche, donner la signature et une définition du prédicat `appartient?` permettant de tester si un nombre n est élément d'un ensemble E.

Par exemple :

`(appartient? 4 (union (singleton 1) (singleton 4)))` → `#t`

`(appartient? 3 (union (singleton 1) (singleton 4)))` → `#f`

`(appartient? 3 (vide))` → `#f`

[3/66]

Quatrième partie

Arbres généraux

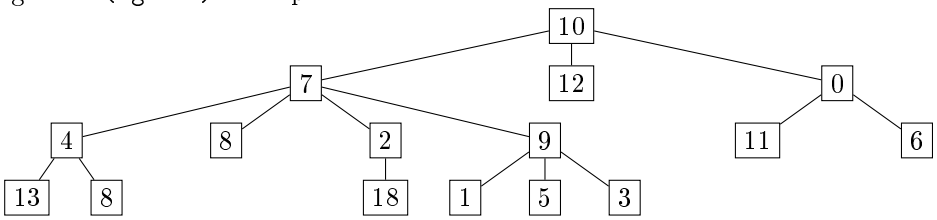
Exercice 7

On s'intéresse dans cet exercice à des arbres généraux d'entiers naturels. On donne à titre d'exemple les deux arbres généraux suivants :

– L'arbre général `(ag-ex1)` constitué simplement d'une feuille :

Numéro d'anonymat :

– L'arbre général `(ag-ex2)` correspondant à :



Question 7.1

À l'aide des accesseurs sur les arbres généraux et les listes, écrire une expression SCHEME permettant de récupérer l'entier `2` à partir de l'arbre donné par `(ag-ex2)`.

[2/66]

Question 7.2

Écrire la signature et une définition de la fonction `ag-somme` qui, étant donné un arbre général `G` contenant des entiers naturels rend la somme des étiquettes présentes dans `G`. Par exemple :

`(ag-somme (ag-ex1))` → `4`

`(ag-somme (ag-ex2))` → `117`

[4/66]

Numéro d'anonymat :