Nom

Prénom

Examen - Avril 2013
UPMC LI101 Vietnam
Durée 2h00
La carte de référence est le seul document autorisé.

Le barème, sur 66 actuellement, est donné à titre indicatif. La note finale sera sur 60, avec des points bonus : le sujet est volontairement long.

Attention : lorsque l'on demande la signature d'une fonction, vous devez aussi donner les hypothèses s'il y en a. Aucune spécification complète n'est demandée.

Vous pouvez utilisez toutes les fonctions de la carte de référence et toute fonction précédemment définie dans l'examen. Sinon, vous devez spécifier et définir toute fonction annexe que vous utilisez.

Remarque : lorsque vous ne trouvez pas la solution d'une question, ne restez pas bloqués, passez à la suite en considérant que vous pouvez utilisez le résultat de la question dans la suite.

Exercice 1

Question 1.1

Donner la signature et une définition de la fonction `maj-note` qui, étant donnée une note x comprise au sens large entre 0 et 20, majore les notes supérieures ou égales à 9,5 de la façon suivante : si $9,5 \leq x < 12$, on ajoute 0,5 ; si $12 \leq x < 15$, on ajoute 1 ; si $15 \leq x < 18$, on ajoute 1,5 ; si $18 \leq x$, on arrondit à 20. Par exemple :

`(maj-note 8)` → `8`

`(maj-note 9.5)` → `10.0`

`(maj-note 12)` → `13`

[2/66]

Nom | Prénom

Question 1.2

Donner la signature et une définition de la fonction `maj-L-notes` qui, étant donnée une liste L de notes comprises au sens large entre 0 et 20, rend la liste obtenue en majorant toutes les notes de L selon la règle énoncée dans la question précédente. Par exemple :

`(maj-L-notes '(8 9.5 12 10 17 13 19 14))` → `(8 10.0 13 10.5 18.5 14 20 15)`

`(maj-L-notes '())` → `()`

[3/66]

Exercice 2

Soit a un entier strictement supérieur à 1, on appellera dans cet exercice *fonction de Syracuse pour* a, la fonction f_a ainsi définie :

$$\forall n \in \mathbb{N} \quad f_a(n) = \begin{cases} n \div a & \text{si } n \text{ divisible par } a \\ n + 1 + (n \div a) & \text{sinon} \end{cases}$$

où $n \div a$ est le quotient de la division euclidienne (division entière) de n par a.

Question 2.1

Calculer $f_2(6)$, $f_2(7)$, $f_3(4)$, $f_3(5)$.

[1/66]

Nom | Prénom

Question 2.2

Donner la signature et une définition de la fonction `syr` qui, étant donnés un entier a strictement supérieur à 1 et un entier naturel n, renvoie $f_a(n)$. Par exemple :

`(syr 2 20)` → `10`

`(syr 2 9)` → `14`

`(syr 3 15)` → `5`

`(syr 3 10)` → `14`

[2/66]

Suite de Syracuse Étant donné un entier a strictement supérieur à 1, la *suite de Syracuse pour* a est définie à partir d'un entier naturel p par :

$$u_0 = p \quad \text{et} \quad \forall n \in \mathbb{N} \quad u_{n+1} = f_a(u_n)$$

Par exemple, la suite de Syracuse pour $a = 2$ définie à partir de $p = 7$ prend les valeurs suivantes : $u_0 = 7$, $u_1 = 11$, $u_2 = 17$, $u_3 = 26$, $u_4 = 13$, $u_5 = 20$, etc.

On appelle *orbite* au rang k de p pour a la suite $(u_k, u_{k-1}, \ldots, u_1, u_0)$ où $(u_n)_{n\in\mathbb{N}}$ est la suite de Syracuse pour a définie à partir de p.

Question 2.3

Donner la signature et une définition de la fonction `orbite` qui, étant donnés un entier a strictement supérieur à 1 et deux entiers naturels p et k, renvoie l'orbite au rang k de p pour a. Veillez à l'efficacité de la fonction ! Par exemple :

`(orbite 2  7 15)` → `(1 2 1 2 1 2 4 8 5 10 20 13 26 17 11 7)`

`(orbite 3  7 15)` → `(47 35 26 19 14 10 7 21 63 47 35 26 19 14 10 7)`

`(orbite 3  7 0)` → `(7)`

Nom

Prénom

[4/66]

Question 2.4

Donner la spécification et une définition de la fonction `max-liste` qui étant donnée une liste non vide de nombres rend la plus grand élément de la liste.

[3/66]

Question 2.5

Donner une définition de la fonction `apogee` qui, étant donnés un entier a strictement supérieur à 1, un entier naturel p et un entier naturel k, renvoie le plus grand nombre apparaissant dans l'orbite au rang k de p pour a. La spécification de la fonction `apogee` est la suivante :

```
;;; apogee : Nat * Nat * Nat -> Nat
;;; (apogee a p k) renvoie  le plus grand nombre apparaissant dans
;;; l'orbite au rang k de p pour a.
```

Nom | Prénom

```
;;; HYPOTHÈSE : a > 1
```

Par exemple :

`(apogee 2 7 15)` → `26`

[2/66]

Question 2.6

On considère la fonction `mys` ainsi définie :

```
;;; mys : Nat * Nat * Nat -> Nat
(define (mys a p k)
  (if (= k 1)
      p
      (max p (mys a (syr a p) (- k 1)))))
```

Dérouler l'appel de `(mys 2 7 4)`. Donner la spécification de `mys`.

[3/66]

Nom | Prénom

Exercice 3

Dans cet exercice, on propose d'implanter des fonctions qui permettent d'obtenir un test de divisibilité original. Ce test repose sur les *rubans de Pascal* qui permettent de calculer, à partir d'un entier n et d'un entier non nul d, un nombre p plus petit que n dont le reste de la division par d est égal au reste de la division de n par d.

Question 3.1

Donner une définition de la fonction `liste-reste` qui étant donnés une liste d'entiers L et un entier non nul d rend la liste des restes de la division euclidienne par d des éléments de L. Si L est de la forme (e_1 e_2 ... e_n), la fonction rend la liste (R(e_1, d) R(e_2,d) ... R(e_n, d)) où R(x,y) est le reste de la division euclidienne de x par y.

Voici sa spécification suivie de quelques exemples :

```
;;;liste-reste : LISTE[int] * int -> LISTE[int]
;;;(liste-reste L d) rend la liste des restes de la division euclienne par d
;;; des éléments de la liste d'origine
;;; HYPOTHESE : d non nul
```

```
(liste-reste (list 2 4 6 8 10) 2) → (0 0 0 0 0)
(liste-reste (list 1 2 3 4 5 6 7 8 9 10) 10) → (1 2 3 4 5 6 7 8 9 0)
(liste-reste (list 3 5 7 9 11) 2) → (1 1 1 1 1)
(liste-reste (list) 4) → ()
(liste-reste (list 3 8 10 12 15 98 102) 10) → (3 8 0 2 5 8 2)
```

[3/66]

Nom	Prénom

Question 3.2

On considère définie la fonction `puiss` :

```
;;; puiss: Nombre * nat  -> Nombre
;;; (puiss x n) rend x^n
(define (puiss x n)
  (if (= n 0)
      1
      (* x (puiss x (- n 1)))))
```

Donner <u>une définition</u> de la fonction `liste-puiss` qui étant donnés un nombre x et un naturel n rend la liste (1 x x^2 ... x^{n-1}) des n premières puissances de x dans cet ordre (la puissance est croissante).

Voici sa spécification :

```
;;; liste-puiss: Nombre * nat -> LISTE[Nombre]
;;; (liste-puiss x n) rend la liste des n premières puissances de x
```

Et voici quelques exemples :

`(liste-puiss 10 5)` → `(1 10 100 1000 10000)`

`(liste-puiss 10 3)` → `(1 10 100)`

`(liste-puiss 10 2)` → `(1 10)`

`(liste-puiss 10 1)` → `(1)`

`(liste-puiss 10 0)` → `()`

[4/66]

Question 3.3

Donner la <u>signature et une définition</u> de la fonction `ruban-pascal` qui étant donnés un entier naturel non nul d et un entier naturel p rend la liste des restes de la division euclidienne des p premières puissances de 10 par d. La liste résultat est donc de la forme (R(10^0,d) R(10^1,d) ... R(10^{p-1},d)).

Une liste ainsi construite est appelée **ruban de Pascal de longueur** p **pour le diviseur** d.

Nom | Prénom

(ruban-pascal 1 5) → (0 0 0 0 0)

(ruban-pascal 2 5) → (1 0 0 0 0)

(ruban-pascal 3 10) → (1 1 1 1 1 1 1 1 1 1)

(ruban-pascal 7 10) → (1 3 2 6 4 5 1 3 2 6)

On pourra utiliser les fonctions précédemment définies.

[2/66]

Question 3.4

Donner <u>une définition</u> de la fonction **nb-chiffre** qui étant donné un entier naturel n rend le nombre de chiffres significatifs dans l'écriture de n en base 10.

Voici la spécification de la fonction suivie de quelques exemples :

```
;;; nb-chiffre : nat -> nat
;;; (nb-chiffre n) rend le nombre de chiffres significatifs dans l'écriture
;;; de l'entier n en base 10
```

(nb-chiffre 4321) → 4

(nb-chiffre 432) → 3

(nb-chiffre 43) → 2

(nb-chiffre 4) → 1

(nb-chiffre 1) → 1

(nb-chiffre 0) → 1

[2.5/66]

Nom | Prénom

Question 3.5

Donner une définition de la fonction `liste-chiffre` qui étant donné un entier naturel n rend la liste des chiffres composant l'écriture de n en base 10. Le chiffre des unités est en tête de la liste résultat.

Voici la spécification de la fonction suivie de quelques exemples :

```
;;; liste-chiffre : nat -> LISTE[nat]
;;; (liste-chiffre n) rend la liste des chiffres de l'écriture de n
;;; en base 10, le chiffre des unités en tête de la liste résultat
```

(liste-chiffre 4321) → (1 2 3 4)

(liste-chiffre 432) → (2 3 4)

(liste-chiffre 43) → (3 4)

(liste-chiffre 4) → (4)

(liste-chiffre 1) → (1)

(liste-chiffre 0) → (0)

[3/66]

Question 3.6

Donner la signature et une définition de la fonction `mult-liste` qui étant données deux listes de nombres $L1$ et $L2$ rend la somme des produits des éléments de L1 et L2 de même rang. On supposera que L1 et L2 sont de même longueur. Si les deux listes sont vides, la fonction rend 0.

Autrement dit, si $L1$ est de la forme (e_1 e_2 ... e_n) et $L2$ de la forme (f_1 f_2 ... f_n) alors le résutlat est : $e_1 * f_1 + e_2 * f_2 + ... + e_n * f_n$

(mult-liste (list 1 4 9) (list 1 10 100)) → 941

(mult-liste (list 4 9) (list 10 100)) → 940

(mult-liste (list 9) (list 100)) → 900

(mult-liste (list) (list)) → 0

Nom

Prénom

[3/66]

Question 3.7

Etant donnés un entier naturel n et un entier naturel non nul d, on souhaite calculer un entier que l'on appellera *nombre-P* dans la suite. Ce nombre-P pour n et d a pour valeur la somme des produits des éléments de la liste des chiffres composant l'écriture de n et des éléments du ruban de Pascal pour d dont la longueur est égale au nombre de chiffres dans l'écriture de n.

Par exemple, on considère 4321 pour n et on choisit pour d la valeur 7. La liste des chiffres de $n = 4321$ est `(1 2 3 4)` et il y a 4 chiffres dans son écriture. Le ruban de Pascal de longueur 4 pour l'entier 7 est égale à (R(1,7) R(10,7) R(100,7) R(1000,7)) soit (1 3 2 6).
Le nombre-P pour n=4321 et d=7 vaut donc 1*1 + 2*3 + 3*2 + 4*6 soit 37.

Donner une définition de la fonction `nombre-P` qui étant donnés un naturel n et un naturel non nul d calcule la valeur du nombre-P pour n et d.

La spécification de la fonction est la suivante :

```
;;; nombre-P: nat * nat -> nat
;;; (nombre-P n d) rend la somme des produits des éléments de la
;;; liste des chiffres composants l'écriture de n et des éléments du ruban
;;; de Pascal dont la longueur est égale au nombre de chiffres dans l'écriture
;;; de n
;;; HYPOTHESE : d est non nul
```

Et voici quelques applications exemples :

`(nombre-P 4321 7)` → 37

`(nombre-P 4321 10)` → 1

`(nombre-P 2 5)` → 2

Remarque : utiliser les fonctions précédemment définies pour répondre à cette question.

Nom

Prénom

[3/66]

Question 3.8

Pour calculer le nombre-P, on parcourt deux fois l'ensemble des chiffres de l'écriture de n en base 10 : une fois pour construire la liste des chiffres, une fois pour calculer le nombre de chiffres dans l'écriture de n. On souhaite ne parcourir qu'une seule fois n en définissant une fonction `liste-longueur` qui étant donné un naturel n rend un couple formé de la liste des chiffres de n et du nombre de chiffres dans l'écriture de n. Par exemple :

`(liste-longueur 1)` → `((1) 1)`

`(liste-longueur 12)` → `((2 1) 2)`

`(liste-longueur 123)` → `((3 2 1) 3)`

`(liste-longueur 1234)` → `((4 3 2 1) 4)`

Donner la signature et une définition de la fonction `liste-longueur`.

[5/66]

Nom

Prénom

Question 3.9

Réécrire une définition de la fonction `nombre-P` qui utilise la fonction `liste-longueur`.

[2/66]

Nom	Prénom

Le but des prochains exercices 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 terme $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$. 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'interesse, 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.

Exercice 4

Dans ce qui suit, on supposera donné 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))
```

Question 4.1

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 :

Nom | Prénom

`(p-nul? (poly2))` → `#f`

`(p-nul? (poly3))` → `#f`

`(p-nul? (polynul))` → `#t`

[1.5/66]

Question 4.2

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]

Question 4.3

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 :

$$a_0 + a_1x + a_2x^2 + \ldots + a_nx^n = a_0 + x * (a_1 + x * (a_2 + x * (\ldots + x * a_n)\ldots))$$

Nom

Prénom

```
(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
```

[3/66]

Question 4.4

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]

Exercice 5

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

Nom | Prénom

des termes correspond à l'addition des coefficents des termes de même degré dans P et P'.

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

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

$$\begin{array}{rr|rrrrrrrr} & P_1 & 14 & 1 & -4 \\ + & P_2 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 5 \\ \hline & P & 14 & 2 & -1 & 0 & 0 & 0 & 0 & 5 \end{array}$$

Question 5.1

Donner <u>une définition</u> 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]

Question 5.2

À l'aide de la fonction précédente, donner <u>la signature</u> et <u>une définition</u> **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.

`(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)`

Nom | Prénom

[3/66]

Question 5.3

Donner <u>une définition</u> de la fonction précédente utilisant une **fonctionnelle**. Cette variante fonctionnelle de la fonction précédente est nommée `p-add-liste-fct` :

`(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)`

[3/66]