Groupe	Nom	Prénom

Devoir sur table - Novembre 2012

LI101

Durée 1h30

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

Le sujet comporte 12 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 : 45 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.

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`

Réponse. [2/45]

```
;;; maj-note : Nombre -> Nombre
;;; (maj-note x) rend la note obtenue en majorant x.
;;; HYPOTHÈSE : 0 <= x <= 20
(define (maj-note x)
  (cond
    ((< x 9.5) x)
    ((< x 12) (+ x 0.5))
    ((< x 15) (+ x 1))
    ((< x 18) (+ x 1.5))
    (else 20)))
```

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 :

Groupe	Nom	Prénom

```
(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 '()) → ()
```

Réponse. [3/45]

```
;;; maj-L-notes : LISTE[Nombre] -> LISTE[Nombre]
;;; (maj-L-notes L) rend la liste obtenue en majorant toutes les notes de L.
;;; HYPOTHÈSE : tous les éléments de L sont compris entre 0 et 20.
(define (maj-L-notes L)
  (if (pair? L)
      (cons (maj-note (car L)) (maj-L-notes (cdr L)))
      (list)))
```

Une autre définition, en utilisant un `map` :

```
(define (maj-L-notes-map L)
  (map maj-note L))
```

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 de n par a.

Question 2.1

Calculer $f_2(6)$, $f_2(7)$, $f_3(3)$, $f_3(4)$, $f_3(5)$.

Réponse. [1/45]
$f_2(6) = 3$, $f_2(7) = 11$, $f_3(3) = 1$, $f_3(4) = 6$, $f_3(5) = 7$.

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
```

Groupe Nom Prénom

Réponse. [2/45]

```
;;; syr : Nat * Nat -> Nat
;;; (syr a n) renvoie la valeur de la fonction de Syracuse pour a calculée en n.
;;; HYPOTHÈSE : a > 1
(define (syr a n)
  (if (= (remainder n a) 0)
        (quotient n a)
        (+ n 1 (quotient n a))))
```

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)`

Réponse. [4/45]

```
;;; orbite : Nat * Nat* Nat -> LISTE[Nat]
;;; (orbite a p k) renvoie l'orbite au rang k de p pour a.
;;; HYPOTHÈSE : a > 1
(define (orbite a p k)
  (if (= k 0)
      (list p)
      (let ((res (orbite a p (- k 1))))
        (cons (syr a (car res)) res))))
```

Le lien avec la conjecture de Syracuse Dans le cas où $a = 2$ on retrouve la fonction de Syracuse bien connue :
$syracuse(n) = n/2$ si n est pair et $syracuse(n) = (3n + 1)/2$ si n est impair,
et la suite de Syracuse définie à partir de p :
$u_0 = p$ et, pour $n > 0$, $u_{n+1} = u_n/2$ si u_n est pair et $u_{n+1} = (3u_n + 1)/2$ si u_n est impair.
La conjecture de Syracuse s'énonce ainsi :

$$\forall u_0 \neq 0 \quad \exists n \in \mathbb{N} \text{ tel que } u_n = 1$$

Dans le cas d'un entier $a > 1$ quelconque, on peut conjecturer que la suite de Syracuse pour a ne prend qu'un nombre fini de valeurs différentes, pour tout u_0. Par exemple, pour $a = 3$ et $u_0 = 7$, la suite $(u_n)_{n\in\mathbb{N}}$ prend les valeurs : $7, 10, 14, 19, 26, 35, 47, 63, 21, 7, 10, 14, 19, 26, 35$, etc. Elle ne prend qu'un nombre fini de valeurs différentes (qui sont $7, 10, 14, 19, 26, 35, 47, 63, 21$).

Groupe	Nom	Prénom

Question 2.4

Donner une <u>définition</u> de la fonction `max-liste` de spécification :

```
;;; max-liste : LISTE[Nombre] -> Nombre
;;; (max-liste L) renvoie le plus grand élément de la liste L.
;;; HYPOTHÈSE : L non vide
```

Réponse. [2/45]

```
(define (max-liste L)
  (if (pair? (cdr L))
      (max (car L) (max-liste (cdr L)))
      (car L)))
```

Question 2.5

Donner une <u>définition</u> 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.
;;; HYPOTHÈSE : a > 1
```

Par exemple :

`(apogee 2 7 15)` → 26

Réponse. [2/45]

```
;;; 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.
;;; HYPOTHÈSE : a > 1
(define (apogee a p k)
  (max-liste (orbite a p k)))
```

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`.

Groupe	Nom	Prénom

Réponse. [3/45]

```
(mys 2 7 4)
(max 7 (mys 2 11 3))
(max 7 (max 11 (mys 2 17 2)))
(max 7 (max 11 (max 17 (mys 2 26 1))))
(max 7 (max 11 (max 17 (max 26 (mys 2 13 0)))))
(max 7 (max 11 (max 17 (max 26 13))))
(max 7 (max 11 (max 17 26)))
(max 7 (max 11 26))
(max 7 26)
26
```

La fonction `mys` a la même spécification que la fonction `apogee`.

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)`

Réponse. [3/45]

```
;;;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 L.
;;; HYPOTHESE : d non nul
(define (liste-reste L d)
  (if (pair? L)
      (cons (remainder (car L) d) (liste-reste (cdr L) d))
      L))
```

Groupe	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 une définition 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)` → `()`

Réponse. [4/45]

```
;;; liste-puiss: Nombre * nat -> LISTE[Nombre]
;;; (liste-puiss x n) rend la liste des n premières puissances de x
;;; dans l'ordre des puissances croissantes.
(define (liste-puiss x n)
  (if (= n 0)
      (list)
      (append (liste-puiss x (- n 1)) (list (puiss x (- n 1))))))
```

Question 3.3

Donner la signature et une définition 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.**

`(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.

Groupe	Nom	Prénom

Réponse. [2/45]

```
;;; ruban-pascal: nat * nat -> LISTE[nat]
;;;(ruban-pascal d p) rend la liste des restes de la division euclidienne des
;;; p premières puissances de 10 par d.
;;; HYPOTHESE : d non nul
(define (ruban-pascal d p)
  (liste-reste (liste-puiss 10 p) d))
```

Question 3.4

Donner une définition 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`

Réponse. [2/45]

```
;;; nb-chiffre : nat -> nat
;;; (nb-chiffre n) rend le nombre de chiffres significatifs dans l'écriture
;;; de l'entier n en base 10
(define (nb-chiffre n)
   (if (< n 10)
    1
    (+ 1 (nb-chiffre (quotient n 10)))))
```

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)`

Groupe Nom Prénom

(liste-chiffre 4) → (4)

(liste-chiffre 1) → (1)

(liste-chiffre 0) → (0)

Réponse. [3/45]

```
;;; 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 premier
(define (liste-chiffre n)
  (if (< n 10)
    (list n)
    (cons (remainder n 10) (liste-chiffre (quotient n 10)))))
```

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

Réponse. [3/45]

```
;;; mult-liste: LISTE[Nombre]*LISTE[Nombre] -> Nombre
;;; (mult-liste L1 L2) rend la somme des multiplications des éléments de L1 et L2 de même rang
;;; HYPOTHESE : les deux listes sont de même longueur
(define (mult-liste L1 L2)
  (if (pair? L1)
      (+ (* (car L1) (car L2)) (mult-liste (cdr L1) (cdr L2)))
      0))
```

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.

Groupe	Nom	Prénom

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.

Réponse. [2/45]

```
;;; nombre-P: nat * nat -> nat
;;; (nombre-P n d) rend la somme des multiplications 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
(define (nombre-P n d)
  (mult-liste (liste-chiffre n) (ruban-pascal d (nb-chiffre n))))
```

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`.

Réponse. [5/45]

```
;;;liste-longueur : nat -> COUPLE[LISTE[nat] nat]
;;;(liste-longueur n) rend le couple formé de la liste des chiffres composant
;;; l'écriture de $n$ en base 10 et le nombre de chiffres dans cette écriture
(define (liste-longueur n)
  (if (< n 10)
    (list (list n) 1)
    (let ((res-rec (liste-longueur (quotient n 10)))
  (reste   (remainder n 10)))
      (list (cons reste (car res-rec)) (+ 1 (cadr res-rec))))))
```

Groupe Nom Prénom

Question 3.9

Réécrire <u>une définition</u> de la fonction `nombre-P` qui utilise la fonction `liste-longueur`.

Réponse. [2/45]

```
;;;nombre-P-bis: nat * nat -> nat
;;;(nombre-P-bis n d) rend la somme des multiplications 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
(define (nombre-P-bis n d)
  (let ((R (liste-longueur n)))
  (mult-liste (car R) (ruban-pascal d (cadr R)))))
```