Groupe	Nom	Prénom

Devoir sur table - Octobre 2010

LI101

Durée 1h30

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

Le sujet comporte 9 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

On suppose que l'on a déjà défini la fonction d'élévation à la puissance dont voici la spécification :

```
;;; puiss : Nombre * nat -> Nombre
;;; (puiss x n) rend x à la puissance n.
```

On veut définir la fonction binaire H d'élévation à l'*hyper puissance* définie de la façon suivante :

$$H(a,n) = \begin{cases} a & \text{si } n = 0 \\ a^{H(a,n-1)} & \text{sinon} \end{cases}$$

Question 1.1

Donner la signature et une définition Scheme de la fonction `hyper-puiss` telle que `(hyper-puiss a n)` vaut $H(\texttt{a},\texttt{n})$.

[3/45]

Groupe	Nom	Prénom

Exercice 2

Soit la suite u_n définie par $\begin{cases} u_0 & = & 5 \\ u_{(n+1)} & = & (n+1)\dfrac{u_n}{2} \end{cases}$

Question 2.1

Donnez la signature et une définition Scheme de la fonction `suite-u` telle que `(suite-u n)` donne la valeur de $u_{\texttt{n}}$.

Exemples :

`(suite-u 0)` → `5`

`(suite-u 1)` → `2.5`

`(suite-u 3)` → `3.75`

[3/45]

Question 2.2

Donnez la signature et une définition Scheme de la fonction `somme-suite-u` telle que `(somme-suite-u n)` donne la valeur de la somme $u_{\texttt{n}} + \ldots + u_0$.

[3/45]

Exercice 3

Le but de cet exercice est de déterminer la liste des n premiers nombres premiers.

Rappels :

– on dit qu'un nombre entier p *divise* un nombre entier n lorsque le reste de la division de n par p est égal à 0 ;

Groupe	Nom	Prénom

– on dit qu'un nombre entier naturel n est *premier* lorsque $n > 1$ et seuls 1 et n divisent n. Par exemple :
  – 7 est premier car seuls 1 et 7 divisent 7 ;
  – 8 n'est pas premier car 2 divise 8.
  Par convention : 1 n'est pas premier.

Question 3.1

Donner une définition de la fonction `carre: Nombre -> Nombre` qui, étant donné un nombre, retourne le carré de ce nombre.

Par exemple :

`(carre 2)` → `4`

`(carre 8)` → `64`

[1/45]

Question 3.2

Donner une définition du prédicat `divise?` dont la spécification est la suivante :

```
;;; divise?: nat * nat -> bool
;;; (divise? p n) retourne #t si p divise n, #f sinon.
;;; HYPOTHESE: p est non nul.
```

Par exemple :

`(divise? 2 8)` → `#t`

`(divise? 3 8)` → `#f`

[2/45]

Question 3.3

Étant donné un nombre m, on veut savoir s'il existe un nombre p (différent de 1 et m) qui divise m.

On rappelle que : si un entier m n'est pas premier, il a au moins un diviseur inférieur ou égal à sa racine carrée.

Groupe	Nom	Prénom

Sans utiliser la fonction prédéfinie `sqrt`, donner une définition du prédicat `existe-diviseur?` dont la spécification est la suivante :

```
;;; existe-diviseur?: nat * nat -> bool
;;; (diviseur? n m) retourne #t si et seulement si m possède un diviseur
;;; compris entre n et la racine carrée de m (au sens large).
;;; HYPOTHESE: n est supérieur ou égal à à 2.
```

Par exemple :

`(existe-diviseur? 2 16)` → `#t` (car 2 divise 16 et $2 < 4 \leq \sqrt{16}$)

`(existe-diviseur? 2 15)` → `#t` (car 3 divise 15 et $3 \leq 5 = \sqrt{15}$)

`(existe-diviseur? 2 17)` → `#f`
(car aucun nombre plus grand que 2 et plus petit que $\sqrt{17}$ ne divise 17)

`(existe-diviseur? 4 9)` → `#f`
(car 4 est supérieur à $3 = \sqrt{9}$ et donc aucun nombre plus grand que 4 et plus petit que 3 n'existe !)

[4/45]

Question 3.4

En utilisant `existe-diviseur?`, donner une définition du prédicat `premier?` dont la spécification est la suivante :

```
;;; premier?: nat -> bool
;;; (premier? n) retourne #t si n est premier, #f sinon.
```

Exemples :

`(premier? 0)` → `#f`

`(premier? 1)` → `#f`

`(premier? 2)` → `#t`

`(premier? 3)` → `#t`

`(premier? 51)` → `#f`

`(premier? 53)` → `#t`

`(premier? 417)` → `#f`

Groupe | Nom | Prénom

[3/45]

Question 3.5

Donner une définition de la fonction `premiers-aux` dont la spécification est la suivante :

```
;;; premiers-aux: nat * nat -> LISTE[nat]
;;; (premiers-aux n m) retourne la liste des m premiers nombres premiers
;;; qui sont supérieurs ou égaux à n
;;; HYPOTHESE: n>=2
```

Par exemple :

`(premiers-aux 2 5)` → `(2 3 5 7 11)`

`(premiers-aux 3 5)` → `(3 5 7 11 13)`

`(premiers-aux 4 5)` → `(5 7 11 13 17)`

`(premiers-aux 5 5)` → `(5 7 11 13 17)`

`(premiers-aux 2 0)` → `()`

`(premiers-aux 422 1)` → `(431)`

[4/45]

Question 3.6

En déduire la signature et une définition de la fonction `premiers` qui retourne la liste des m premiers nombres premiers.

Groupe | Nom | Prénom

[2/45]

Exercice 4

Un météorologue amateur qui a suivi l'UE LI101 utilise des listes pour mémoriser ses relevés de température. Il procède chaque jour à 2 relevés, matin et soir, qu'il mémorise dans une liste des relevés mensuels constituée d'une liste de relevés hebdomadaires.

On a les types suivants :
T1 `LISTE[Nombre]` : liste à deux éléments des relevés quotidiens ;
T2 `LISTE[LISTE[Nombre]]` : liste des relevés quotidiens ;
T3 `LISTE[LISTE[LISTE[Nombre]]]` : liste des relevés hebdomadaires.
On écrira `LISTE^3[Nombre]` comme abréviation de `LISTE[LISTE[LISTE[Nombre]]]`.

La procédure de renseignement de cette petite base de données comporte essentiellement 3 actions :
A1 chaque début de semaine, on rajoute une liste vide pour la nouvelle semaine en fin de la liste du dernier mois ;
A2 chaque soir, on rajoute dans la liste de la semaine courante (la dernière de la liste du mois) la liste contenant les deux valeurs relevées ce jour.
Dans cet exercice, nous allons définir quelques fonctions de gestion des données recueillies. Pour cela, nous aurons besoin de quelques fonctions générales sur les listes.

Question 4.1

Donner la signature et une définition de la fonction `ajout-fin` qui ajoute un élément en fin de liste.

Exemples :

```
(ajout-fin '(12.3 17.1) (list)) → ((12.3 17.1))
```

```
(ajout-fin '(12.3 17.1) '((10.2 15.0) (12.7 15.9)))
→ ((10.2 15.0) (12.7 15.9) (12.3 17.1))
```

[3/45]

Groupe	Nom	Prénom

Question 4.2

Donnez la spécification et une définition de la fonction `creer-semaine` qui crée une nouvelle semaine (vide) dans la liste mensuelle (*cf* action A1).

Exemples :

`(creer-semaine (list))` → `(())`

`(creer-semaine '('((10.2 15.0) (12.7 15.9) (12.3 17.1) ...) ) )` → `( ( (10.2 15.0) (12.7 15.9) (1`

[2/45]

Question 4.3

Voici la spécification de la fonction `ajout-jour` qui réalise l'action A2 :

```
;;; ajout-jour: Nombre * Nombre * LISTE^3[Nombre] -> LISTE^3[Nombre]
;;; (ajout-jour t1 t2 mois) ajoute la liste contenant t1 et t2
;;; comme dernier relevé pour la liste mois (en fin de la dernière
;;; semaine du mois).
;;; HYPOTHESE: mois contient au moins une semaine.
```

Exemples (pour alléger, les semaines ne sont pas complètes) :

`(ajout-jour 5.7 12.5 '((list) ) )` → `((list (5.7 12.5)))`

`(ajout-jour 5.7 12.5 '(((4.8 13.2) ) ) )` → `(((4.8 13.2) (5.7 12.5)))`

`(ajout-jour 5.7 12.5 '(((4.8 13.2) (6.1 12.9) ) (list) ) )`
→ `(((4.8 13.2) (6.1 12.9)) ((5.7 12.5)))`

`(ajout-jour 5.7 12.5 '(((4.8 13.2) (6.1 12.9) ) ((5.1 14.0) ) ) )`
→ `(((4.8 13.2) (6.1 12.9)) ((5.1 14.0) (5.7 12.5)))`

Donner une définition de la fonction `ajout-jour`.

[5/45]

Groupe	Nom	Prénom

Exercice 5

Cet exercice prolonge l'exercice précédent, mais l'on cherche ici à exploiter les données météorologiques d'un mois mémorisées dans une liste de type `LISTE^3[Nombre]`. Il s'agira de calculer des moyennes et des amplitudes de variations de températures.

Question 5.1

Donner la signature et une définition de la fonction `somme-jour` qui calcule la somme des deux relevés d'une journée.

Exemple :

`(somme-jour '(7.2 17.3))` → `24.5`

[2/45]

Question 5.2

Donnez la signature et une définition de la fonction `somme-semaine` qui calcule la somme des relevés d'une semaine (*cf* type T2).

Exemple (ici encore, pour alléger, la semaine n'est pas complète) :

`(somme-semaine '((3.4 13.6) (3.2 15.1) (4.5 16.4)))` → `56.2`

[3/45]

Question 5.3

On suppose définie la fonction `length` dont voici la spécification

```
;;; length: LISTE[alpha] -> nat
;;; (length L) donne la longueur (nombre d'éléments) de la liste L.
```

Donner une définition de la fonction `nb-jours` qui compte le nombre de relevés effectués dans un mois : c'est-à-dire, la somme des nombres de jours dans chaque semaine. Le nombre de jours d'une semaine est simplement la longueur de la liste correspondante.

Groupe | Nom | Prénom

Exemples :

`(nb-jours (list))` → `0`

`(nb-jours '(((3.4 13.6) (3.2 15.1) (4.5 16.4))))` → `3`

`(nb-jours '(((3.4 13.6) (3.2 15.1) (4.5 16.4)) ((6.2 17.3))))` → `4`

[3/45]

Question 5.4

On suppose définie la fonction `somme-mois` dont voici la spécification :

```
;;; somme-mois: LISTE^3[Nombre] -> Nombre
;;; (somme-mois mois) calcule la somme des températures
;;; relevées dans la liste mois. Donne 0 si le mois est vide.
```

Donner une définition de la fonction `moyenne-mois` qui calcule la moyenne des relevés d'un mois.

[2/45]

`*current-note*` → `45`