Groupe	Nom	Prénom

Devoir sur table - Novembre 2009
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 : 44 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

Le but de cet exercice est de déterminer une valeur approchée du nombre π à l'aide du développement suivant :

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \ldots + \frac{1}{n^2} + \ldots = \lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2}$$

Soit $s(n)$ la somme finie $\sum_{k=1}^{n} \frac{1}{k^2} = 1 + \frac{1}{2^2} + \ldots + \frac{1}{n^2}$ définie pour tout entier naturel strictement positif n. Nous dirons que cette somme est une valeur approchée de $\frac{\pi^2}{6}$ au rang n.

Question 1.1

Donner <u>une définition</u> Scheme de la fonction `s: nat -> Nombre` telle que `(s n)` donne la valeur de $1 + \frac{1}{2^2} + \ldots + \frac{1}{n^2}$ pour tout entier naturel strictement positif n.

[2.5/44]

Groupe Nom Prénom

Question 1.2

En déduire <u>une définition</u> Scheme de la fonction `approx-pi: nat -> Nombre` telle que `(approx-pi n)` donne la valeur approchée de π de rang n pour tout entier naturel strictement positif n.

[1.5/44]

Exercice 2

Nous désirons définir la fonction `liste-meme-taille` qui étant données deux listes `L1` et `L2` teste si les deux listes sont de même longueur. Voici la spécification de la fonction :

```
;;;liste-meme-taille : LISTE[alpha] * LISTE[alpha] -> bool
;;;(liste-meme-taille L1 L2) teste si L1 et L2 sont de même longueur
```

Question 2.1

Avant de définir la fonction, nous voulons établir un jeu de tests qui nous permettra de nous assurer que la définition sera correcte. Selon les longueurs respectives des arguments `L1` et `L2`, si ces longueurs sont positives ou nulles, il existe plusieurs manières pour que la valeur de `liste-meme-longueur` soit `#f` ou `#t`.

Donner un jeu de tests, c'est-à-dire un ensemble d'applications permettant de couvrir l'ensemble des possibilités listées ci-dessus.

[3/44]

Groupe	Nom	Prénom

Question 2.2

Voici une définition pour la fonction `liste-meme-taille` qui comporte une erreur.

```
(define (liste-meme-taille L1 L2)
  (if (and (pair? L1) (pair? L2))
      (liste-meme-taille (cdr L1) L2)
      (and (not (pair? L2)) (not (pair? L1)))))
```

On cherche à mettre en évidence l'erreur du code précédent. Pour cela, donner une application de `liste-meme-taille` qui renvoie un résultat incorrect ainsi que quelques étapes de l'évaluation de cette application permettant de mettre en évidence l'erreur. Il est recommandé de s'appuyer sur les applications du jeu de tests de la question précédente pour répondre à cette question.

[2/44]

Question 2.3

Proposer une définition Scheme correcte de la fonction `liste-meme-taille`.

[1/44]

Exercice 3

Question 3.1

Donner la signature et une définition de la fonction `decompose` qui étant donné un entier naturel `n` rend la liste des chiffres apparaissant dans l'écriture de `n`. Le chiffre des unités de `n` est le premier de la liste retournée.

Par exemple :

Groupe	Nom	Prénom

`(decompose 567)` → `(7 6 5)`

`(decompose 56)` → `(6 5)`

`(decompose 5)` → `(5)`

`(decompose 0)` → `(0)`

`(decompose 3)` → `(3)`

[4/44]

Question 3.2

Donner la signature et une définition *récursive* de la fonction `renverse` qui étant donnée une liste `L` rend la liste composée des éléments de `L` mais dans le sens inverse.

Par exemple :

`(renverse (list 1 2 3 4))` → `(4 3 2 1)`

`(renverse (list "je" "tu" "il"))` → `("il" "tu" "je")`

`(renverse (list))` → `()`

[3/44]

Question 3.3

Donner la signature et une définition *récursive* de la fonction `liste-egale?` qui étant données deux listes `L1` et `L2` teste si les deux listes sont égales c'est-à-dire sont composées des mêmes éléments et ces éléments sont dans le même ordre.

Groupe	Nom	Prénom

Par exemple :

`(liste-egale? (list 2 3 4) (list 1 2 3 4))` → `#f`

`(liste-egale? (list  "je" "tu" "il") (list "je" "tu" "il"))` → `#t`

`(liste-egale? (list  "je" "tu" "elle") (list "je" "elle" "tu"))` → `#f`

`(liste-egale? (list "je" "tu") (list))` → `#f`

`(liste-egale? (list) (list "je" "tu"))` → `#f`

`(liste-egale? (list) (list))` → `#t`

[4/44]

On dit qu'un entier naturel est un palindrome si sa lecture de gauche à droite donne le même entier que sa lecture de droite à gauche. Par exemple, 454 et 3 sont des palindromes alors que 100 n'en est pas un.

Question 3.4

Donner la signature et une définition du prédicat `palindrome?` qui étant donné entier naturel `n` teste si `n` est un palindrome.

Par exemple :

`(palindrome? 454)` → `#t`

`(palindrome? 4)` → `#t`

`(palindrome? 1001)` → `#t`

`(palindrome? 100)` → `#f`

Indication : On peut remarquer que si un nombre n est un palindrome, alors la liste L des chiffres apparaissant dans l'écriture de n est identique à la liste renversée de L.

Groupe Nom Prénom

[2/44]

Question 3.5

Donner la signature et une définition du prédicat `liste-palindrome` qui étant donnés deux entiers naturels `n` et `m` tels que `n` $\leq$ `m` rend la liste des entiers compris entre `n` et `m` inclus qui sont des palindromes.

Par exemple :

`(liste-palindrome 0 10)` $\rightarrow$ `(0 1 2 3 4 5 6 7 8 9)`

`(liste-palindrome 20 35)` $\rightarrow$ `(22 33)`

`(liste-palindrome 100 140)` $\rightarrow$ `(101 111 121 131)`

[3/44]

Exercice 4

Le but de cet exercice est de permettre le calcul d'une valeur approchée de la fonction mathématique cos. On s'appuiera pour cela sur le fait que $\cos(x)$ peut être défini comme la limite, lorsque n tend vers l'infini, de la somme suivante[1] :

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + (-1)^n \frac{x^{2n}}{(2n)!} + \ldots = \lim_{n \to \infty} \sum_{k=0}^{n} (-1)^k \frac{x^{2k}}{(2k)!}$$

[1] le réel x désigne ici un angle exprimé en radian.

Groupe Nom Prénom

Question 4.1

Donner une définition Scheme de la fonction `puiss: Nombre * nat -> Nombre` telle que `(puiss x n)` donne la valeur de x^n.

[2/44]

Pour les prochaines questions, on pourra utiliser la fonction `puiss` qui vient d'être définie ainsi que la fonction `fact` dont la spécification est la suivante :

```
;;; fact: nat -> nat
;;; (fact n) calcule la factorielle de l'entier n
```

Soit $x \in \mathbb{R}$ un nombre réel. On définit récursivement la suite $u_n(x)$ pour $n \in \mathbb{N}$ par :

$$\begin{cases} u_0(x) & = & 1 \\ u_n(x) & = & u_{n-1}(x) + \dfrac{(-1)^n x^{2n}}{(2n)!} \quad \text{pour } n \geq 1 \end{cases}$$

Question 4.2

Donner la signature et une définition Scheme de la fonction `u` telle que `(u x n)` donne $u_n(x)$.

[3/44]

Question 4.3

En utilisant la fonction précédente, donner la signature et une définition de la fonction `u-liste` qui, pour un réel `x` et un entier naturel `n`, donne la liste contenant $u_n(x) \ldots u_0(x)$, dans cet ordre.

Groupe	Nom	Prénom

Exemples :

`(u-liste pi 0)` → `(1)`

`(u-liste pi 3)` → `(-1.2113528429825005 0.1239099258720886 -3.934802200544679 1)`

[2/44]

La précédente fonction est particulièrement inefficace puisque celle-ci recalcule tous les produits intermédiaires liés aux puissances et factorielles pour chacun des éléments de la liste. Un raisonnement sur la récurrence donnée précédemment permet cependant de déduire une définition équivalente de $u_n(x)$ que nous nommerons $v_n(x)$ et qui permet d'éviter ces calculs redondants :

$$\begin{cases} v_0(x) & = & 1 \\ v_1(x) & = & 1 - \frac{x^2}{2} \\ v_n(x) & = & v_{n-1}(x) - (v_{n-1}(x) - v_{n-2}(x)) \cdot \frac{x^2}{2n(2n-1)} \quad \text{pour } n \geq 2 \end{cases}$$

Question 4.4

Donner la signature et une définition Scheme de la fonction `v-liste` telle que `(v x n)` donne la liste contenant $v_n(x) \ldots v_0(x)$ dans cet ordre.

[4/44]

Groupe Nom Prénom

Exercice 5

Un *algorithme de tri* est une procédure qui prend une liste de nombres et renvoie la liste contenant ces nombres en ordre croissant. L'exercice suivant a pour but d'implanter une variante de l'algorithme de **tri par insertion**.

Cet algorithme repose sur la constatation suivante. Lorsque l'on veut trier une liste, deux cas sont possibles. Soit la liste est vide (et donc triée !). Soit la liste n'est pas vide et peut donc se décomposer en un premier élément e et le reste de la liste R. L'algorithme de tri par insertion consiste alors à trier R dans un premier temps puis à insérer e au bon endroit dans cette liste triée dans un second temps.

Question 5.1

Donner la signature et une définition Scheme de la fonction `insertion-triee` qui étant donnés un nombre `x` et une liste `L` dont les éléments sont rangés en ordre croissant rend la liste `L` dans laquelle `x` a été ajouté de telle sorte que les éléments sont toujours rangés en ordre croissant.

Par exemple :

`(insertion-triee 3 (list 1 2 4 5))` → `(1 2 3 4 5)`

`(insertion-triee 3.5 (list 4.5 8.0))` → `(3.5 4.5 8.0)`

`(insertion-triee 3 (list 1 2 3 4))` → `(1 2 3 3 4)`

`(insertion-triee 47 (list 13 20 45))` → `(13 20 45 47)`

`(insertion-triee 47 (list))` → `(47)`

[3/44]

Question 5.2

À l'aide de la fonction précédente, donner une signature et une définition Scheme de la fonction `tri-insertion` qui étant donnée une liste de nombres `L` rend la liste `L` triée selon l'algorithme décrit ci-dessus.

[4/44]