Université de La Réunion | lundi 5 juin 2000

DEUG 1ere année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal

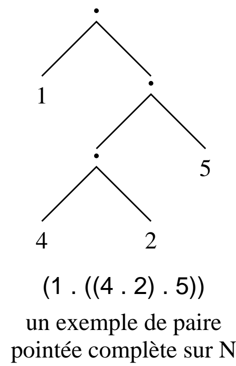
Durée : 2 heures – Aucun document n'est autorisé. 1

Répondre uniquement dans les cadres prévus à cet effet

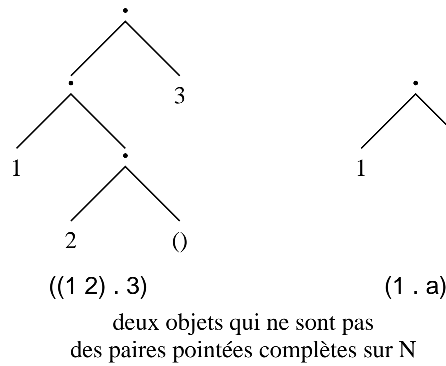
Nom :	N° étudiant :	Signature
Prénom(s) :		
Date de naissance :		

Partie 1 : Paires pointées et listes

Définition 1 : Une paire pointée multi–niveaux est dite *complète sur* $\mathbb{N}$ si toutes les feuilles de sa représentation arborescente (où car est represénté par le sous–arbre gauche et cdr est représenté par le sous–arbre droit) appartiennent à $\mathbb{N}$.
Le schéma suivant montre les représentations arborescentes d'un exemple de paire pointée complète sur $\mathbb{N}$ et de deux contre-exemples.



(1 . ((4 . 2) . 5))
un exemple de paire pointée complète sur N



((1 2) . 3) (1 . a)
deux objets qui ne sont pas des paires pointées complètes sur N

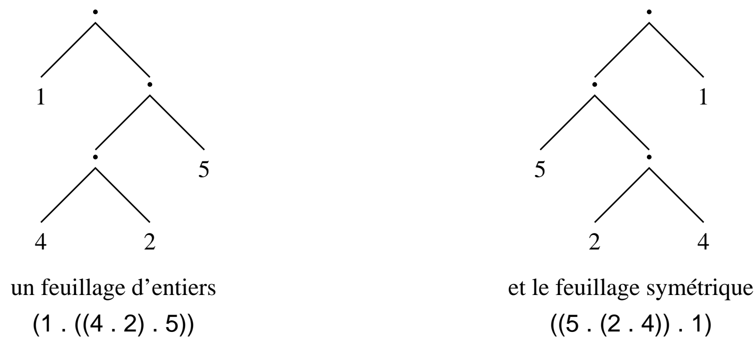
Définition 2 : On appelle *feuillage d'entiers* un objet qui est, soit un entier, soit une paire pointée complète sur $\mathbb{N}$.

(4 pts) **Exercice 1 :** Écrivez en Scheme un prédicat feuillage? qui teste si un objet est un feuillage d'entiers.

Tournez SVP ⟹

(3 pts) **Exercice 2 :** Écrivez en Scheme une fonction liste-feuilles qui prend en argument un feuillage d'entiers et renvoie la liste des feuilles de sa représentation arborescente (de gauche à droite).
Par exemple `(liste-feuilles '(1 . ((1 . 2) . 2)))` doit renvoyer `(1 1 2 2)`.

(3 pts) **Exercice 3 :** Écrivez en Scheme une fonction sym qui prend en argument un feuillage d'entiers et renvoie le feuillage d'entiers symétrique. Par exemple `(sym '(1 . ((4 . 2) . 5)))` doit renvoyer `((5 . (2 . 4)) . 1)`.



Université de La Réunion lundi 5 juin 2000

DEUG 1ere année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal

Durée : 2 heures – Aucun document n'est autorisé. **2**

Répondre uniquement dans les cadres prévus à cet effet

Nom :	N° étudiant :	Signature
Prénom(s) :		
Date de naissance :		

Partie 2 : Type abstrait de données

On définit, au moyen des fonctions suivantes, le type abstrait de données $\mathcal{S}$ qui permet de manipuler les suites strictement croissantes finies d'entiers naturels :

– les fonctions génératrices : $\begin{cases} \textbf{suite-vide} & : & & \rightarrow & \mathcal{S} \\ \textbf{ajouter-terme} & : & \mathcal{S} \times \mathbb{N} & \rightarrow & \mathcal{S} \end{cases}$

– les fonctions caractéristiques : $\begin{cases} \textbf{est-vide?} & : & \mathcal{S} & \rightarrow & \{\#f, \#t\} \\ \textbf{min} & : & \mathcal{S} & \rightarrow & \mathbb{N} \\ \textbf{enlever-min} & : & \mathcal{S} & \rightarrow & \mathcal{S} \end{cases}$

Exercice 1 : En supposant que l'on code une suite croissante à l'aide d'une liste dont les éléments sont rangés par ordre croissant, écrivez les fonctions génératrices et caractéristiques données ci-dessus.

*(0,5 pt)*1a. La fonction **suite-vide** qui ne prend pas d'argument et renvoie une suite vide :

*(0,5 pt)*1b. La fonction **est-vide?** qui teste si une suite est vide ou pas :

*(0,5 pt)*1c. La fonction **min** qui renvoie le plus petit terme d'une suite :

*(0,5 pt)*1d. La fonction **enlever-min** qui retire d'une suite son plus petit terme.

Tournez SVP ⟹

(2 pts) 1e. La fonction **ajouter-terme**, qui fonctionne comme suit: pour tout $S \in \mathcal{S}$ et tout $x \in \mathbb{N}$, (**ajouter-terme** S x) ajoute à la suite S le nouveau terme x. Attention, le résultat doit être une suite strictement croissante.

Exercice 2 : Toutes les fonctions écrites dans cet exercice doivent être indépendantes de l'implantation du type abstrait de données. Vous les définirez en utilisant les fonctions génératrices et caractéristiques mentionnées plus haut.

(3 pts) 2a. Ecrivez la fonction **union** qui prend en argument deux suites croissantes $S1$ et $S2$ et qui renvoie une suite croissante dont les termes sont l'union de ceux de $S1$ et $S2$.

Exemple : l'union des suites 1, 2, 4, 6, 9 et 1, 3, 5, 9 est la suite 1, 2, 3, 4, 5, 6, 9.

(3 pts) 2b. Ecrivez la fonction **intersection** qui prend en argument deux suites croissantes $S1$ et $S2$ et qui renvoie une suite croissante dont les termes sont l'intersection de ceux de $S1$ et $S2$.

Exemple : l'intersection des suites 1, 2, 4, 6, 9 et 1, 3, 6, 9 donne la suite 1, 6, 9.

Université de La Réunion lundi 5 juin 2000

DEUG 1ere année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal

Durée : 2 heures – Aucun document n'est autorisé. **3**

Répondre uniquement dans les cadres prévus à cet effet

Nom :	N° étudiant :	Signature
Prénom(s) :		
Date de naissance :		

Partie 3 : Complexité en temps

Remarque importante : Dans les exercices qui suivent, vous préciserez clairement en fonction de quelle mesure des données vous évaluez les complexités en temps. Vous expliciterez tout aussi clairement les équations de récurrence permettant de calculer ces complexités et après les avoir résolues vous donnerez leur ordre de grandeur en utilisant la notation O.
On vous demande d'évaluer la complexité en temps dans le pire des cas. C'est à dire, en fonction d'une mesure des données, le temps maximum consommé pour évaluer un appel à la fonction.

(4 pts) **Exercice 1 :** Évaluer la complexité en temps dans le pire des cas de la fonction suivante qui prend comme paramètres un objet et une liste plate.

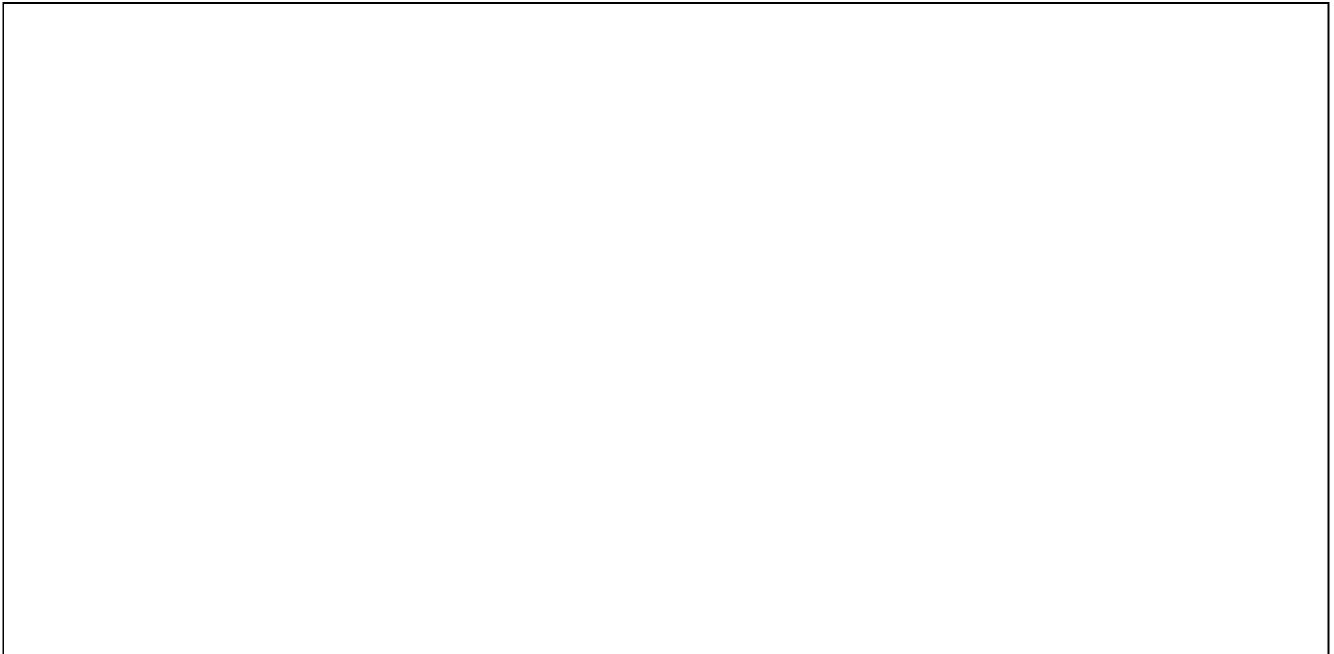
```
; estdans : Objet, Liste --> Booleen
;              x ,   l   |-> #t si x est dans l
;                            #f sinon

(define estdans
   (lambda (x l)
      (cond ((null? l) #f)
            ((equal? x (car l)) #t)
            (else (esdans x (cdr l))))))
```

Tournez SVP ⟹

(6 pts) **Exercice 2 :** Évaluer la complexité en temps dans le pire des cas de la fonction suivante qui prend comme paramètres deux listes plates.

```
; truc : Liste, Liste --> Liste
;          l1 ,   l2  |-> remplace tous les elements de l1 qui sont dans l2 par des 1
;                         et tous les element de l1 qui ne sont pas dans l2 par des 0
(define truc
   (lambda (l1 l2)
       (cond ((null? l1) '())
             ((estdans (car l1) l2) (cons 1 (truc (cdr l1) l2)))
             (else (cons 0 (truc (cdr l1) l2))))))
```



Université de La Réunion lundi 5 juin 2000

DEUG 1ere année – MIAS 1 et MASS 1

UE4 – Informatique 2 – Examen terminal

Durée : 2 heures – Aucun document n'est autorisé. 4

Répondre uniquement dans les cadres prévus à cet effet

Nom : N° étudiant : Signature
Prénom(s) :
Date de naissance :

Partie 4 : Ordre supérieur

(4 pts) **Exercice 1 :** Donnez le résultat de l'évaluation des expressions suivantes :

```
>(apply append '((1) ((a)) (a(b(c)))))

>(map cdr '((1 2) (2 (3 a))))

>(map car '((2 3 4) () (a (b))))

>(map * '(1 2) '(5 6) '(3 2))

>(map (lambda (x) (or (odd? x) (>0 x))) '(-7 9 4 0 ))

>((lambda (a b c) ( / (* a a ) (- b c) ) ) 4 6 4 )

>(apply * (map (lambda (x) 2) '(2 (1 1) 0 (4 (6 5))) ))

>(apply + (map (lambda (l) (+ (cadr l) 1) ) '((1 2) (2 3) (3 4))))
```

(3 pts) **Exercice 2 :** Ecrivez, en utilisant la fonction prédéfinie `map`, une fonction non récursive `remplacer` qui prend comme paramètre une liste plate d'entiers `l` et qui remplace tous les entiers pairs de `l` par 0 et tous les entiers impairs par 1.

Par exemple `'(remplacer '(1 2 3 4 5 6 7 8))` doit renvoyer la liste `'(1 0 1 0 1 0 1 0)`.

Tournez SVP ⟹

(3 pts) **Exercice 3 :** En utilisant les fonctions prédéfinies `apply` et `map`, écrire une fonction `moyennes`, non récursive, qui prend comme paramètre une liste `ll` de listes de nombres et renvoie la liste des moyennes des nombres de chaque liste présente dans `ll`.

Par exemple : `(moyennes '((10 20) (12 14 16 14) (8 10 12)) )` doit renvoyer la liste `'(15 14 10)`.