



**Licence S&T 1<sup>o</sup> Année, UE INF121**  
**Informatique – Devoir Surveillé**

**Durée 2h, sans documents**

**12 Mars 2005**

Il est demandé de respecter le lexique des énoncés. **NE PAS LES RECOPIER**. Le barème donné est indicatif. Les exercices sont indépendants. Répondre **UNIQUEMENT** sur la feuille prévue à cet effet. *Les solutions seront données en utilisant la notation fonctionnelle du cours et des travaux dirigés (et non CAML)*. Les termes "définition récursive" et "définition récurrente" sont équivalents.

**Rappel des notations concernant les séquences**

- Constructeurs : [], S•e, e o S ; testeur et sélecteurs : EstVide?, début, dernier, premier, fin.
- La concaténation est notée par le symbole &.

**1. Vérification des types des noms d'une expression algébrique [3 points]**

On s'intéresse ici à la vérification des types des noms qui apparaissent dans une expression algébrique par rapport aux spécifications des opérations qu'elles mettent en jeu. Par exemple, pour que l'expression **(3 o a)•premier(b)** soit correcte du point de vue des types, les contraintes suivantes doivent être respectées: **a** doit être de type séquence d'entiers et **b** de type séquence non vide d'entiers. Si ces conditions sont respectées, l'expression **(3o a)•premier(b)** est de type séquence non vide d'entiers.

**Q1.**

— Pour chacune des expressions nommées E1, E2, E3 et E4 données ci-dessous, donner les contraintes de types que doivent respecter les noms y apparaissant, puis le type de l'expression si ces contraintes sont respectées.

E1 : si (si 3<4 alors 4<3 sinon 3=4) = a alors (b et a) sinon (a et b)

E2 : premier (soit x=c+1 dans [x, d])

E3 : < e=(f=3), [f] >

E4 : [ si g<3 alors [] sinon "o" o h ]

**2. A propos de paragraphes et de questionnaires [10 points]**

Dans cet exercice on considère des phrases, des paragraphes, ainsi que la notion de questionnaire. Pour définir précisément ces concepts, on donne le lexique suivant.

Ponctuation : le type caractère { *restreint aux caractères "?", "!" et "."* }

Lettre : le type caractère { *restreint aux lettres de l'alphabet* }

Mot : le type séquence non vide de Lettres

Expression : le type séquence non vide de Mots

Phrase : le type <une Expression, une Ponctuation>

Question : le type <une Expression, "?">

Affirmation : le type <une Expression, une Ponctuation { *restreint aux caractères "!" et "."* } >

Paragraphe : le type séquence non vide de Phrases  
 Texte : le type séquence de caractères  
 ph\_cvt : la Phrase < ["comment","vas","tu"], "?" >  
 ph\_qft : la Phrase < ["que","fais","tu"], "?">  
 ph\_jt : la Phrase < ["je","travaille"],"! ">  
 ph\_tb : la Phrase < ["très","bien"],".">  
 ph\_cef : la Phrase < ["c","est","fini"],"! ">  
 ph\_qheti : la Phrase < ["quelle","heure","est","il"],"?">  
 ph\_9h30 : la Phrase < ["neuf","heure","trente"],".">  
 pa1 : le Paragraphe [ph\_cvt,ph\_tb,ph\_qft,ph\_jt,ph\_qheti,ph\_9h30]  
 pa2 : le Paragraphe [ph\_jt,ph\_jt,ph\_cef,ph\_tb,ph\_tb]  
 pa3 : le Paragraphe [ph\_jt,ph\_9h30,ph\_cef,ph\_tb]  
 pa4 : le Paragraphe [ph\_cvt]  
 pa5 : le Paragraphe [ph\_tb, ph\_cef]  
 pa6 : le Paragraphe [ph\_cvt,ph\_jt,ph\_qheti,ph\_jt]  
 pa7 : le Paragraphe [ph\_cvt,ph\_jt,ph\_qheti]

TexteDePhrase : une Phrase → un Texte

*{ TexteDePhrase(PH) est le texte formé de la séquence des mots de PH séparés par des tirets et terminé par la ponctuation de PH. Il n'y a pas de tiret après le dernier mot. Par exemple TexteDePhrase(<["que","fais","tu"],"?">) = "que-fais-tu?" }*

TexteDeParagraphe : un Paragraphe → un Texte

*{ TexteDeParagraphe(PA) est le texte formé de la séquence des phrases du paragraphe séparées par des tirets.*

*Par exemple TexteDeParagraphe(pa5) = "très-bien.-c-est-fini!" }*

EstQuestion : une Phrase → un booléen

*{ EstQuestion(PH) est vrai si et seulement si PH est une question.*

*Par exemple EstQuestion(ph\_cvt) = vrai EstQuestion(ph\_jt) = faux }*

EstAffirmation : une Phrase → un booléen

*{ EstAffirmation(PH) est vrai si et seulement si PH est une affirmation.*

*Par exemple EstAffirmation(ph\_9h30) EstAffirmation(ph\_cef) }*

Ponctue : une Phrase, une Ponctuation → une Phrase

*{ Ponctue(PH,PO) est la phrase formée de l'expression de PH mais ayant PO comme ponctuation.*

*Par exemple Ponctue(<["ca","va"],"?">,"!") = <["ca","va"],"! "> }*

EstQuestionnaire : un Paragraphe → un booléen

*{ EstQuestionnaire(PA) est vrai si et seulement si le paragraphe PA est une séquence alternée de question - affirmation. Autrement dit un questionnaire débute par une question, chaque question est immédiatement suivie d'une affirmation, deux affirmations ne peuvent pas se suivre. Un questionnaire est donc une séquence de longueur paire dans laquelle les éléments de rangs impairs sont des questions et les éléments de rangs pairs sont des affirmations.*

*Par exemple*

*EstQuestionnaire(pa1) = vrai      EstQuestionnaire(pa2) = faux*

*EstQuestionnaire(pa4) = faux      EstQuestionnaire(pa6) = vrai*

*EstQuestionnaire(pa7) = faux }*

**Q2.**

— Donner le type puis la valeur de l'expression suivante.

$E : \text{soit } x = \text{Ponctue}(\text{ph\_jt}, "?") \circ (\text{ph\_jt} \circ (\text{Ponctue}(\text{ph\_tb}, "?") \circ [\text{Ponctue}(\text{ph\_tb}, "!") ]))$   
dans  $\langle \text{TexteDeParagraphe}(x), \text{EstQuestionnaire}(x) \rangle$

— Donner une réalisation de la fonction Ponctue.

— Donner une réalisation de la fonction EstQuestion.

— Donner une réalisation de la fonction EstAffirmation en utilisant la fonction EstQuestion.

**Q3.**

— Donner une définition récursive de la fonction TexteDeParagraphe en se basant sur la fonction TexteDePhrase (2 équations).

— Donner une définition récursive de la fonction TexteDePhrase sans utiliser de fonction intermédiaire (2 équations).

— En se basant sur cette définition récursive, donner une réalisation récursive pour cette fonction en se basant sur une traduction la plus systématique possible.

**Q4.**

— Donner une définition récursive de la fonction EstQuestionnaire sans utiliser d'autres fonctions que celles spécifiées dans cet exercice (3 équations)..

### **3. Addition d'entiers naturels définis récursivement [8 points]**

La récursivité est un mécanisme très général. Il a été montré en cours que l'ensemble des entiers naturels pouvait être "construit" de manière récursive selon plusieurs modèles. Deux modèles ont été présentés. Dans cet exercice on s'intéresse au premier modèle qui a été appelé  $E_n$ . On rappelle que  $\mathbb{N}$  est l'ensemble défini récursivement à partir de la valeur constante notée 0 et du constructeur nommé  $s$ . L'ensemble  $\mathbb{N}$  est donc formé des éléments 0,  $s(0)$ ,  $s(s(0))$ , etc. Il existe clairement une correspondance entre l'ensemble  $\mathbb{N}$  et l'ensemble des entiers positifs. De même, il a été montré en cours comment définir les opérations arithmétiques sur l'ensemble  $\mathbb{N}$ . On donne le lexique suivant :

$\mathbb{N}$ VersEP : un  $\mathbb{N} \rightarrow$  un entier  $\geq 0$

*{ NVersEP(A) est l'entier correspondant à A. Par exemple NVersEP(s(s(0)))=2 }*

EPVers $\mathbb{N}$  : un entier  $\geq 0 \rightarrow$  un  $\mathbb{N}$

*{ EPVersN(A) est l'élément de  $\mathbb{N}$  correspondant à A.*

*Par exemple EPVersN(2) = s(s(0)) }*

Add : deux  $\mathbb{N} \rightarrow$  un  $\mathbb{N}$

*{ Add(A,B) est l'élément de  $\mathbb{N}$  correspondant à la somme des entiers naturels correspondant à A et B. Par exemple Add(s(s(0))),s(0)) = s(s(s(0))) }*

$$E1 : \text{EPVersN}(0+2)$$

$$E2 : \text{NVersEP}(\text{s}(\text{EPVersN}(0)))+1$$

$$E3 : \text{EPVersN}(\text{NVersEP}(\text{s}(\text{s}(\text{s}(0)))) + 2 )$$

$$E4 : \text{Add}(\text{s}(0), \text{Add}(\text{EPVersN}(1), \text{EPVersN}(1)))$$

**Q5.**

- Donner la valeur des expressions E1, E2, E3 et E4
- Donner une définition récursive de la fonction nommée NVersEP (2 équations).
- Donner une définition récursive de la fonction nommée EPVersN (2 équations).
- Donner une réalisation récursive de la fonction nommée EPVersN

**Q6.**

— Compléter les équations récursives ci-dessous sachant que chaque équation doit permettre "d'avancer" dans le processus de calcul récursif de l'addition.

1.  $\text{Add}( 0 , 0 ) = \underline{\hspace{4cm}}$

2.  $\text{Add}( x , 0 ) = \underline{\hspace{4cm}}$

3.  $\text{Add}( 0 , y ) = \underline{\hspace{4cm}}$

4.  $\text{Add}(\text{s}(x) , 0 ) = \underline{\hspace{4cm}}$

5.  $\text{Add}( 0 , \text{s}(y) ) = \underline{\hspace{4cm}}$

6.  $\text{Add}( x , \text{s}(y) ) = \text{s}( \underline{\hspace{3cm}} )$

7.  $\text{Add}(\text{s}(x) , y ) = \text{s}( \underline{\hspace{3cm}} )$

8.  $\text{Add}(\text{s}(x) , \text{s}(y) ) = \underline{\hspace{2cm}} \text{Add}(x,y) \underline{\hspace{2cm}}$

**Q7.**

Parmi les combinaisons d'équations ci-dessous, certaines ne sont pas des définitions récursives. De telles combinaisons ne permettent pas de calculer l'addition pour tout couple d'entier naturels.

— Indiquer pour chaque combinaison s'il s'agit d'une définition récursive et sinon donner un exemple d'addition ne pouvant être calculé.

C1 : équations 1 et 8

C2 : équations 3 et 7

C3 : équations 4, 5 et 8

**Q8.**

On rappelle que le testeur et le sélecteur correspondant au modèle  $E_n$  sont respectivement noté  $p$  (pour précédent) et  $\text{EstNul?}$ . Ainsi on a les propriétés suivantes

$$\text{EstNul?}(0)$$

$$\text{non EstNul?}(\text{s}(x))$$

$$p(\text{s}(x)) = x$$

— Donner une réalisation récursive de la fonction Add.