Grenoble 1

UFR IMA
Informatique & Mathématiques Appliquées

UNIVERSITE
JOSEPH FOURIER
Sciences, Technologie, Médecine

MIAS 1°A, UE SM22 ; SM 2°A, module H7 ; STPI 2°A/IUP1 Miage

Informatique – Examen, 1ère session

Durée 3h, sans documents **4 Mai 2000**

Il est demandé de respecter le lexique des énoncés. *NE PAS LES RECOPIER.* Le barème donné est indicatif. Les problèmes sont indépendants.

1. A propos d'ensembles

Rappel des notations concernant les séquences

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

La *différence* de deux ensembles **X** et **Y**, habituellement notée **X \ Y**, est l'ensemble des éléments de **X** n'appartenant pas à **Y**.

On considère des ensembles d'entiers représentés sous forme de séquences d'entiers distincts deux à deux, et on spécifie une fonction nommée **Dif**, de la manière suivante :

EnsEnt : le type séquence d'entiers *{les entiers de la séquence sont distincts deux à deux}*
Dif : deux EnsEnt → un EnsEnt *{Dif(X, Y) = X \ Y}*

Q1.[4 points]. Traiter **chacune** des deux analyses suivantes de la fonction **Dif.**

(i) *Aucun ordre n'est imposé sur les séquences représentant les ensembles* (données et résultat).

Donner des équations de récurrence définissant la fonction **Dif** en termes d'une fonction intermédiaire adéquate que l'on spécifiera et dont on donnera aussi des équations de récurrence.

(ii) *Les séquences représentant les ensembles sont triées en ordre croissant* (données et résultat).

Donner des équations de récurrence définissant directement la fonction **Dif** (sans fonction intermédiaire) en exploitant le fait que les séquences (données et résultat) sont triées.

Indication : appliquer le *principe d'interclassement* de séquences vu en cours.

2. Eléments de rang multiple de R

Etant donné une séquence **S**, on veut construire la sous-séquence de **S** formée des éléments de **S** dont le rang dans **S** est un multiple d'un entier **R** donné. On convient que le rang du premier élément est 1 ; le rang d'un élément autre que le premier est 1 + le rang de son prédécesseur.

Q2. [5 points]

(i) On spécifie les fonctions suivantes :

Elément : un type ; SéqE : le type séquence d'Eléments
LesRangsImpairs : une SéqE → une SéqE
{LesRangsImpairs(S) est la séquence des éléments de rang impair dans S.
LesRangsImpairs([]) = [].
*Par exemple, LesRangsImpairs([**8**, 2, **3**, 5, **2**, 13]) = [8, 3, 2].}*
LesRangsPairs : une SéqE → une SéqE
*{LesRangsPairs(S) est la séquence des éléments de rang pair dans S ou la séquence vide s'il n'y en a pas. Par exemple, LesRangsPairs([8, **2**, 3, **5**, 2, **13**]) = [2, 5, 13].}*

— Donner des équations de récurrence définissant la fonction **LesRangsImpairs**.
— Donner une réalisation de la fonction **LesRangsPairs** en termes de la fonction **LesRangsImpairs**.

(ii) On spécifie une fonction nommée **ElémentsDeRang** qui généralise ce qui précède :

ElémentsDeRang : un entier > 0, une SéqE → une SéqE

{ElémentsDeRang(R, S) est la séquence des éléments de S dont le rang est multiple de R ou la séquence vide s'il n'y en a pas.
Par exemple,ElémentsDeRang(3, [8, 2, ***3****, 5, 2,* ***13****, 4]) = [3, 13]}*

Pour décrire la fonction **ElémentsDeRang** on introduit une fonction **LesDerniers** :

LesDerniers : un entier ≥ 0, une SéqE → une SéqE

{LesDerniers(X, S) est la séquence construite à partir de S en enlevant ses X premiers éléments. En particulier, LesDerniers(0, S) = S. Si la longueur de la séquence est inférieure ou égale à X, le résultat est la séquence vide.
Par exemple, LesDerniers(2, [8, 2, 3, 5, 2, 13, 4]) = [3, 5, 2, 13, 4].}

— Donner des équations de récurrence définissant la fonction **LesDerniers**.
— Donner une **réalisation récursive** de la fonction **ElémentsDeRang** en utilisant la fonction **LesDerniers**.

3. Des motifs réguliers

On appelle *motif simple*, une séquence d'entiers de longueur impaire, formée de deux entiers en alternance, comme le montrent les exemples suivants :

`[8]`	`[8,9,8]`	`[8,9,8,9,8]`	`[8,9,8,9,8,9,8]`	• • •
(1)	(3)	(5)	(7)	

Un *motif simple* est caractérisé par sa longueur, forcément impaire (entre parenthèses en dessous des exemples), les deux entiers la constituant (8 et 9 dans les exemples) et l'entier par lequel elle commence (8 dans les exemples).

Par ailleurs, on appelle *motif double* une séquence de motifs simples de la forme suivante :

```
(1)   [ [8] ]
(3)   [ [8,9,8],[9],[8,9,8] ]
(5)   [ [8,9,8,9,8],[9,8,9],[8],[9,8,9],[8,9,8,9,8] ]
(7)   • • •
```

Un *motif double* est caractérisé par le nombre (forcément impair) de motifs simples qu'il comporte (entre parenthèses à gauche des exemples), les deux entiers les constituant (8 et 9 dans les exemples) et le premier entier de son premier motif simple (8 dans les exemples). Le premier élément d'un motif double de **n** éléments est un motif simple de **n** entiers. Le premier entier de chaque motif simple alterne d'un motif simple au suivant. La taille des motifs simples décroît dans un premier temps de 2 en 2 puis croît de 2 en 2.

La structure d'un motif double est illustrée, sur un exemple, par la présentation suivante :

```
(5)  [ [ 8,9,8,9,8 ],
         [ 9,8,9 ],
           [ 8 ],
         [ 9,8,9],
       [ 8,9,8,9,8] ]
```

Dans ce qui suit, on étudie une fonction de construction d'un motif double.

Q3.[3 points]

On spécifie une fonction nommée **MotifDouble** de la manière suivante :

MotifDouble : un entier impair > 0, deux entiers →
une séquence non vide de (séquences non vides d'entiers)

{MotifDouble(n, e1, e2) est le motif double de n motifs simples constitués des entiers e1 et e2, et dont le premier motif simple commence par e1.
Par exemple, MotifDouble(5, 8, 9) = [[8,9,8,9,8], [9,8,9], [8], [9,8,9], [8,9,8,9,8]]}

— Spécifier une fonction intermédiaire nommée **MotifSimple** de construction d'un motif simple et donner des équations de récurrence la définissant.
— Donner des équations de récurrence définissant la fonction **MotifDouble**.

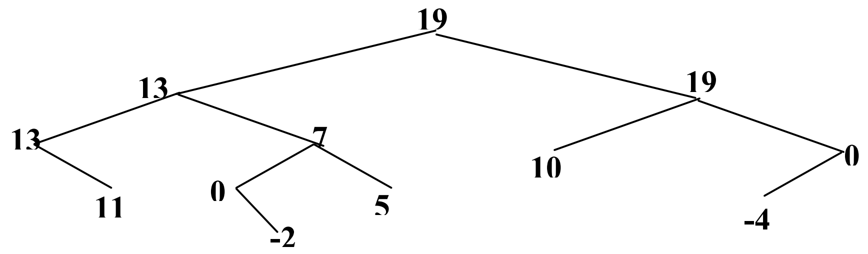
4. Arbres binaires partiellement ordonnés

Rappel des notations concernant les arbres binaires

— Constructeurs : /\, /G, r, D\ ; testeur et sélecteurs : EstVide?, Racine, Gauche, Droit.

— Pour les arbres binaires non vides, on peut utiliser la notation abrégée : //r\\, //G,r\\, //r, D\\, //G, r, D\\

Un arbre binaire d'entiers est *partiellement ordonné*, si tout élément de l'arbre a une valeur supérieure ou égale aux valeurs de chacun de ses deux fils (s'ils existent). On convient qu'un arbre vide est partiellement ordonné.



Un arbre binaire d'entiers partiellement ordonné

On étudie quelques fonctions portant sur un arbre partiellement ordonné. Puis on étudie un algorithme de tri fondé sur l'utilisation d'un tel arbre.

Les questions Q4 à Q7 peuvent être traitées dans un ordre quelconque après les avoir toutes lues.

Q4. **[3 points] Conséquences de la définition des arbres partiellement ordonnés**

(i) Quelques propriétés d'un arbre partiellement ordonné

— Que peut-on dire des sous-arbres d'un arbre partiellement ordonné?

— La *valeur d'un chemin* de l'arbre est la séquence des entiers associés à chacun des nœuds de ce chemin, ceux-ci étant énumérés en parcourant le chemin de père en fils.
Donner une propriété vérifiée par la valeur d'un chemin de l'arbre, quel que soit ce chemin.

— Que peut-on dire de la valeur associée à la racine d'un arbre partiellement ordonné.

(ii) Un arbre est-il partiellement ordonné

On spécifie la fonction suivante :

EstPO : un arbre binaire d'entiers → un booléen

{EstPO(A) est vrai ⇔ A est un arbre partiellement ordonné.}

Donner des équations de récurrence définissant la fonction **EstPO**.

Q5.[1 point] Construction d'un arbre partiellement ordonné

On étudie la construction d'un arbre partiellement ordonné à partir d'une séquence d'entiers. On spécifie à cet effet une fonction nommée **UnArbrePO** :

AbPO : le type arbre binaire d'entiers *{partiellement ordonné}*

UnArbrePO : une séquence d'entiers → un AbPO

{UnArbrePO(S) est un arbre partiellement ordonné formé de tous les éléments de S et seulement eux. UnArbrePO([]) = /\.}

On dispose d'une fonction nommée **Plus**, spécifiée de la manière suivante :

Plus : un AbPO, un entier → un AbPO

{Plus(A, E) construit un arbre binaire partiellement ordonné comportant E et tous les éléments présents dans A, et seulement eux.}

Donner des équations de récurrence définissant la fonction **UnArbrePO**, en utilisant la fonction **Plus** (on ne demande pas de réaliser la fonction **Plus**).

Q6. [2 points] Suppression de la racine

On spécifie une fonction nommée **SaufRac** :

SaufRac : un AbPO non vide → un AbPO

{SaufRac(A) construit un arbre binaire partiellement ordonné comportant tous les éléments présents dans A, sauf la racine de A.}

Compléter les équations de récurrence suivantes, pour définir la fonction **SaufRac** :

(1) SaufRac(//r\\) = • • • • • • • •
(2) SaufRac(//G, r\\) = • • • • • • • •
(3) SaufRac(//r, D\\) = • • • • • • • •
(4) SaufRac(//G, r, D\\) = • • • • • • • •

*Lors de la correction, on appréciera les solutions n'utilisant pas de fonction intermédiaire (en particulier la fonction **Plus**) et exploitant les propriétés des arbres partiellement ordonnés.*

Q7. [2 points] Un principe de tri

On veut réaliser le tri d'une séquence d'entiers en exploitant la structure d'arbre partiellement ordonné : dans un premier temps, on construit un arbre partiellement ordonné à partir d'une séquence ; dans un deuxième temps, on produit une séquence en ordre croissant des entiers de l'arbre. On spécifie une fonction nommée **TriParArbrePO** :

TriParArbrePO : une séquence d'entiers → une séquence d'entiers en ordre croissant

{TriParArbrePO(S) est une séquence en ordre croissant formée de tous les éléments de S et seulement eux. La réalisation procède selon le principe énoncé ci-dessus.}

— Déterminer les fonctions nécessaires pour appliquer le principe ci-dessus et spécifier celles qui n'ont pas été étudiées dans les questions précédentes.
— Donner une réalisation de la fonction **TriParArbrePO** en utilisant ces fonctions.
— Donner des équations de récurrence définissant la ou les fonctions introduites.