



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, 1^{ère} session**Durée 3h, sans documents****5 Mai 2001**

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

Les solutions seront données en utilisant la notation fonctionnelle du cours et des travaux dirigés (*et non CAML*).

Rappel des notations concernant les séquences

- Constructeurs : [], S•e, e₀S ; testeur et sélecteurs : EstVide?, début, dernier, premier, fin.
- La concaténation est notée par le symbole &.

Rappel des notations concernant les arbres

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

1. Rang du dernier positif

On spécifie une fonction nommée **RDP** (Rang du Dernier Positif) de la manière suivante :

RDP : une séquence d'entiers → un entier ≥ 0

{RDP(S) est le rang dans S du dernier élément strictement positif de S. Par exemple, RDP([12, -1, 23, 0, 10, 35, -5, **23**, 0, -2]) = 8 (le deuxième exemplaire de 23 est le 8^{ème} élément de S). Si S ne comporte aucun élément positif, RDP(S) = 0.}

Q1. [3 points]

On veut donner deux définitions récurrentes de la fonction **RDP**, la première en isolant le dernier élément (découpage à droite), la deuxième en isolant le premier élément (découpage à gauche).

Pour cela compléter les équations de récurrence ci-dessous, en remplaçant les points de suspension (••••). Le cas échéant, on spécifiera et on donnera des équations de récurrence définissant toute fonction intermédiaire jugée nécessaire :

version 1 RDP([]) = 0	version 2 RDP([]) = 0
RDP(S•e) = •••••	RDP(e ₀ S) = •••••

2. A propos de "pangrammes"

Un *pangramme* est un texte qui comporte au moins un exemplaire de chacune des 26 lettres de l'alphabet, sans tenir compte de la différence entre minuscules et majuscules. Voici des exemples :

Portez ce vieux whisky au juge blond qui fume.

Voyez le brick géant que j'examine près du wharf.

Voyez ce bon fakir moqueur pousser un wagon en jouant du xylophone.

The brown quick fox jumps over the lazy dog.

On considère un texte sans lettres majuscules : on veut déterminer si c'est un pangramme. On spécifie à cet effet une fonction nommée **EstPangramme** :

EstPangramme : un texte → un booléen

{EstPangramme(T) a la valeur vrai ⇔ T est un pangramme.

Pré-condition : T ne comporte pas de lettres majuscules}

Rappel : une *pré-condition* est une propriété que **doivent** respecter les données d'une fonction lors de son utilisation. Lors de la description d'une fonction (équations ou réalisation), on supposera que les pré-conditions sont vérifiées.

Pour réaliser la fonction **EstPangramme**, on spécifie une fonction intermédiaire :

AuMoinsUn : un texte non vide, un texte \rightarrow un booléen

{*AuMoinsUn* (*T1*, *T2*) a la valeur vrai \Leftrightarrow pour tout caractère *c* du texte *T1*, il existe dans *T2* au moins un exemplaire de *c*. **Pré-condition** : les éléments de *T1* sont distincts 2 à 2.}

Q2. [4 points]

— Donner des équations de récurrence définissant la fonction **AuMoinsUn** et le cas échéant la ou les fonctions intermédiaires introduites.

— Donner une réalisation de la fonction **EstPangramme**, en utilisant la fonction **AuMoinsUn**.

3. Élagage d'un arbre

(en français, "élagage = action d'enlever les branches superflues (inutiles) d'un arbre")

On s'intéresse à l'*élagage* d'un arbre binaire *au niveau n* : il s'agit d'enlever de cet arbre tous les nœuds de niveau strictement supérieur à *n*. On rappelle que dans un arbre, le niveau de la racine est **1** et que si *n* est le niveau d'un nœud de l'arbre, les fils éventuels de ce nœud sont de niveau *n+1*.

Q3. [3 points]

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

Élagage : un arbre binaire d'Éléments, un entier $> 0 \rightarrow$ un arbre binaire d'Éléments

{*Élagage*(*A*, *n*) est l'arbre *B* obtenu à partir de *A* en ne retenant que les nœuds de niveau inférieur ou égal à *n*. Si *A* est vide, *B* l'est aussi.}

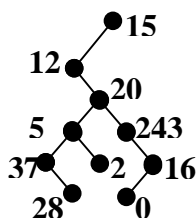
— Donner des équations de récurrence définissant la fonction **Élagage**.

4. A propos de chemins d'accès aux nœuds d'un arbre binaire

Le *chemin d'accès* à un nœud dans un arbre binaire est le chemin menant de la racine à ce nœud.

On choisit de représenter un chemin d'accès à un nœud *N* par une séquence de *k* bits (entiers valant 0 ou 1) où *k* est le nombre **d'arcs** séparant la racine du nœud *N*. Chaque bit est associé à un arc du chemin : s'il vaut 0, l'arc relie un nœud à son fils gauche ; s'il vaut 1, l'arc relie un nœud à son fils droit. Les bits apparaissent dans l'ordre correspondant au chemin de la racine vers le nœud *N*. Enfin, on convient que la séquence vide décrit le chemin conduisant de la racine à elle-même.

Un exemple est donné dans la Figure 1 ci-dessous : on y trouve un arbre d'entiers et un tableau donnant pour chaque entier de l'arbre, le chemin d'accès correspondant.



15 \rightarrow	[]	12 \rightarrow	[0]
20 \rightarrow	[0,1]	5 \rightarrow	[0,1,0]
37 \rightarrow	[0,1,0,0]	28 \rightarrow	[0,1,0,0,1]
2 \rightarrow	[0,1,0,1]	243 \rightarrow	[0,1,1]
16 \rightarrow	[0,1,1,1]	0 \rightarrow	[0,1,1,1,0]

Figure 1 : chemins d'accès aux nœuds d'un arbre binaire

Dans ce qui suit, on fixe le lexique suivant :

Bit : le type entier sur [0, 1]

Chemin : le type séquence de Bits

Élément : un type

Q4. [5 points]

(i) Recherche d'un élément

On veut déterminer si une séquence de bits *S* correspond à un chemin d'accès à l'un des nœuds d'un arbre **ARB** et, si c'est le cas, on veut connaître la valeur de l'élément auquel cette séquence permet d'accéder.

On spécifie pour cela la fonction suivante :

RechParChemin : un arbre binaire d'Éléments, une séquence de bits

→ un booléen, un Élément

{posons $\langle T, Val \rangle = \text{RechParChemin}(ARB, S)$:

– Si la séquence de bits S correspond à un chemin de l'arbre ARB , T a la valeur vrai et Val a la valeur de l'élément accessible par le chemin S .

– Sinon, T a la valeur faux et la valeur de Val n'est pas spécifiée.

Par exemple, si ARB est l'arbre de la Figure 1, $\text{RechParChemin}(ARB, [0,1,0,1]) = \langle \text{vrai}, 2 \rangle$, $\text{RechParChemin}(ARB, [0,1,1,0,0]) = \langle \text{faux}, ? \rangle$ (le ? dénote une valeur quelconque d'élément).}

— Donner des équations de récurrence définissant la fonction **RechParChemin**.

(ii) Chemin d'accès à un élément.

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

ChAccès : un Élément, un arbre binaire d'Eléments, → un booléen, un Chemin

{Posons $\langle T, C \rangle = \text{ChAccès}(Elt, ARB)$.

– S'il existe un élément de valeur Elt dans l'arbre ARB , T a la valeur vrai et C est le chemin d'accès à cet élément.

– Sinon, T a la valeur faux et la valeur de C n'est pas spécifiée.

Pré-condition : les éléments de ARB sont distincts 2 à 2.

Par exemple, si ARB est l'arbre de la Figure 1, $\text{ChAccès}(243, ARB) = \langle \text{vrai}, [0,1,1] \rangle$, $\text{ChAccès}(1042, ARB) = \langle \text{faux}, ? \rangle$.

— Donner des équations de récurrence définissant la fonction **ChAccès**.

5. A propos de sous-séquences de longueur k

Étant donnée une séquence S , une *sous-séquence* de S est une séquence obtenue à partir de S en enlevant un nombre quelconque d'éléments de S et en conservant l'ordre initial dans S pour les éléments restants.

Une sous-séquence de S est *connexe*, si tous ses éléments sont adjacents dans S .

Un *préfixe* de S est une sous-séquence connexe de S commençant par le premier élément de S .

Toute séquence S est sous-séquence d'elle-même.

Par exemple, si $S = [23, 41, -5, 18, 23, 0]$, alors $[23, -5, 23]$ est une sous-séquence (non connexe) de S ; $[41]$ et $[18, 23]$ sont des sous-séquences (connexes) de S ; $[-5, 41]$ n'est pas une sous-séquence de S (l'ordre initial n'est pas respecté) ; $[23, 41, -5]$ est un préfixe de S , mais $[41, -5]$ n'en est pas un (ne commence pas par le premier élément de S).

Dans ce qui suit, on utilise le lexique suivant :

Élément : un type

SéqE : le type séquence d'Eléments

Q5. [6 points]

(i) Le préfixe de longueur k

Pour une séquence S de longueur n et k un entier positif inférieur ou égal à n , il y a exactement un préfixe de longueur k . On spécifie une fonction nommée **LePréfixe** :

LePréfixe : une SéqE **non vide**, un entier > 0 → une SéqE

{ $\text{LePréfixe}(S, k)$ est le préfixe de S de longueur k .

Pré-condition : k est inférieur ou égal à la longueur de S .}

— Donner des équations de récurrence définissant la fonction **LePréfixe**.

(ii) Les sous-séquences connexes de longueur k

— Soit une séquence **S** de longueur **n** et un entier **k** strictement positif.

Donner en fonction de **n** et de **k** le nombre de sous-séquences connexes de longueur **k** dans **S**. Justifier la réponse.

— On spécifie une fonction nommée **LesSousSéqC** de la manière suivante :

LesSousSéqC : une SéqE **non vide**, deux entiers $> 0 \rightarrow$ une séquence de (SéqE non vides)

{LesSousSéqC(S, n, k) est une séquence formée des sous-séquences connexes de longueur k de la séquence S, sachant que n est la longueur de S. Si $k > n$, LesSousSéqC(S, n, k) = [].}

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

$$(1) \text{LesSousSéqC}([e], 1, k) = \bullet \bullet \bullet \bullet$$

$$(2) \text{LesSousSéqC}(e_0S, n+1, k) = \bullet \bullet \bullet \bullet$$

$$\{n > 0, S \text{ non vide}\}$$

Indication : utiliser la fonction **LePréfixe**.

(iii) Les sous-séquences de longueur k

On s'intéresse maintenant à **toutes** les sous-séquences de longueur **k**, qu'elles soient connexes ou non.

— Donner une séquence **S** de 5 éléments et toutes ses sous-séquences de longueur 3.

— Soit une séquence **S** de longueur **n** et un entier **k** strictement positif : donner en fonction de **n** et de **k** le nombre de sous-séquences (connexes ou non) de longueur **k** dans **S**. Justifier la réponse.

— On spécifie une fonction nommée **LesSousSéq** de la manière suivante :

LesSousSéq : une SéqE **non vide**, un entier $> 0 \rightarrow$ une séquence de (SéqE non vides)

{LesSousSéq(S, k) est une séquence formée des sous-séquences (connexes ou non) de longueur k de S. Si k est strictement supérieur à la longueur de S, LesSousSéq(S, k) = [].}

Donner des équations de récurrence définissant la fonction **LesSousSéq**. Le cas échéant, on spécifiera et on donnera des équations de récurrence définissant toute fonction intermédiaire jugée nécessaire.