Université Joseph Fourier | UFR IMA | Département Scientifique Universitaire

LICENCE 1ère année : UE INF121

ALGORITHMIQUE ET PROGRAMMATION FONCTIONNELLE

Interrogation de TP — Groupe INMA7

Sébastien Laborie

mars 2005

Durée : 40 min.

Sans documents ; appareils électroniques interdits.

Barème indicatif : **1** : 7,5 points ; **2** : 12,5 points

Les questions sont indépendantes. Les solutions seront écrites en CAML. Répondre **uniquement** sur les « feuilles de réponses » fournies. L'utilisation de l'ordinateur est laissée à votre appréciation.

1 Lecture d'expressions CAML

Si l'on soumet l'expression `let a=4 in a+premier(b)` à l'interpréteur CAML, on obtient le message :

```
>let a=4 in a+premier(b)
>
L'indentificateur b n'est pas défini.
```

Pour que cette expression puisse être acceptée par CAML, il est nécessaire de spécifier `b`. Du point de vue du type, il faut que `b` soit une séquence non vide d'entiers ; du point de vue de la valeur, prenons par exemple `[1]`. L'expression `let a=4 in a+premier(b)` est alors de type `int` et sa valeur est `5` :

```
#let b=[1] ;;
b : int list = [1]
#let a=4 in a+premier(b) ;;
- : int = 5
```

Q1. Chacune des expressions du tableau 2 de la question Q1 page 4 est incorrecte au sens précisé ci-dessus. Donner les définitions CAML (`let ...= ...`) les rendant correctes. Préciser alors le type et la valeur de l'expression.

2 À propos de majorité

On cherche à écrire en CAML une fonction appelée lesMajeurs dont une spécification est :

> lesMajeurs : une séquence d'entiers → une séquence d'entiers
> {*lesMajeurs(séq) est l'ensemble (sans doublon) des âges supérieurs à 18 ans de séq. Par exemple, lesMajeurs ([16;18;20;17;16;17;18]) = [18;20]*}

Q2. Proposer page 4 un jeu de tests significatifs. Pour chaque test, donner également la réponse attendue.

L'étudiant Pierre Weis propose de réaliser lesMajeurs en CAML de la façon suivante :

```
let rec (lesMajeurs : int list  → int list) = function
    []  → 0
  | e::s  → (if (e >= 18)
             then (if ap(e, lesMajeurs(s)) then [] else [e])
             else []) @ lesMajeurs(s)
```

où la spécification de la fonction ap est :

> ap : un élément, une séquence d'éléments → un booléen
> {*ap(x, séq)=vrai si et seulement si* $x \in$ *séq*}

Q3. La réalisation de Pierre Weis est-elle correcte ? Si non, quel est le message d'erreur de l'interpréteur CAML ? Expliquer **précisément** page 5 d'où vient ce message d'erreur et proposer une correction.

L'étudiant Xavier Leroy, plus malin, n'a pas commis l'erreur de son collègue. En revanche, il a effectué les appels récursifs sur `e::s` au lieu de `s`.

Q4. Quel est le message d'erreur renvoyé par l'interpréteur CAML lorsque Xavier Leroy teste sa réalisation de lesMajeurs sur `[16;17]` ? Expliquer **précisément** page 5 la signification de ce message.

On suppose maintenant que la fonction lesMajeurs ne comporte plus d'erreur. On désire examiner son exécution.

Q5. Indiquer page 5 la commande CAML permettant de tracer les exécutions de lesMajeurs. Présenter sous une forme indentée la trace correspondant à l'appel `lesMajeurs([18;17;18])`. Combien y-a-t'il d'appels récursifs ?

Q6. Cette trace présente-t-elle des appels redondants ? Si oui, les encadrer, et proposer une réalisation permettant de les supprimer. Donner alors la trace de `lesMajeurs([18;17;18])` et le nombre d'appels récursifs.

Feuilles de réponses : corrigé

Groupe : _____

Nom : _____ Prénom : _____

7,5 **Q1.**

	expression	définitions	type	valeur
1,5	let (p,r,z) = x in if p then r*2 else a - int_of_float(z)	let x = (true,1,2.0) (bool*int*float) let a = 1	int	2
1,5	fin([("A",a);(b,'B')])	let a='A' let b="B"	(string*char) list	[("B",'B')]
1,5	let t=premier(x @ debut(y)) in t = (t+.t)/.t	let x = [1.0] (* float list *) let y = [2.3;4.5]	bool	false
3	ajd(fin(Racine(A)@premier(debut(x))), y)	let A = abNV(abV,[1;2;3],abV) let x = [[2;4;5];[4;5]] let y = 3	'a list ('a=int)	[2;3;2;4;5;3]

2 **Q2.**

	test	réponse
0,5	[]	[]
0,5	[17]	[]
0,5	[18]	[18]
0,5	[16;17;18;19;18;20;17;16]	[18;19;20]

2 **Q3.** Réalisation de Pierre Weis :

Elle est incorrecte du point de vue des types: `(if (e >= 18) ... else []) @ lesMajeurs(s)` : "cette expression est de type int list, mais est utilise avec le type int". Le profil de la fonction indique que la valeur retournee doit etre de type int list, ce qui est bien le cas ici. En revanche, pour le cas de base, la valeur retournee est de type int. D'ou le message d'erreur.
Correction : il suffit de remplacer 0 par [].

1,5 **Q4.** Réalisation de Xavier Leroy :

"Exception non rattrape: Out of memory". L'appel recursif etant toujours effectue sur la meme valeur, CAML met en attente les calculs intermediaires sans jamais pouvoir converger vers un cas de base, et sature la memoire de l'ordinateur.

3,5 **Q5.**

Commande trace lesMajeurs : trace("lesMajeurs")

Trace de `lesMajeurs([18;17;18])` :

```
lesMajeurs ⟵ [18; 17; 18]
  lesMajeurs ⟵ [17; 18]
    lesMajeurs ⟵ [18]
      lesMajeurs ⟵ []
      lesMajeurs ⟶ []
      lesMajeurs ⟵ [] redondance
      lesMajeurs ⟶ []
    lesMajeurs ⟶ [18]
  lesMajeurs ⟶ [18]
  lesMajeurs ⟵ [17; 18] redondance
    lesMajeurs ⟵ [18] redondance
      lesMajeurs ⟵ [] redondance
      lesMajeurs ⟶ []
      lesMajeurs ⟵ [] redondance
      lesMajeurs ⟶ []
    lesMajeurs ⟶ [18]
  lesMajeurs ⟶ [18]
lesMajeurs ⟶ [18]
- : int list = [18]
9 appels recursifs
```

3,5 **Q6.**

```
let rec (lesMajeurs : int list ⟶ int list) = function
    []   ⟶ []
  | e::s ⟶ let lesMajeursS = lesMajeurs(s)
           in (if (e ≥ 18)
               then (if ap (e, lesMajeursS) then [] else [e])
               else []) @ lesMajeursS
```

```
lesMajeurs ⟵ [18; 17; 18]

    lesMajeurs ⟵ [17; 18]

        lesMajeurs ⟵ [18]

            lesMajeurs ⟵ []

            lesMajeurs ⟶ []

        lesMajeurs ⟶ [18]

    lesMajeurs ⟶ [18]

lesMajeurs ⟶ [18]

- : int list = [18]

4 appels recursifs
```