

Deug MIAS 1^{ère} année : UE SM23

Examen de TP Informatique

1^{ère} session

26 avril 2002

Durée : 2h00.

Barème indicatif : **1** : 3,5 points ; **2** : 8 points ; **3** : 8,5 points

Les problèmes, et dans une large mesure les questions, sont indépendants.

Sans documents ni calculatrice.

Respecter le lexique des énoncés. Ne pas les recopier.

Numérotez vos réponses de 1 à 13, conformément à l'énoncé.

Rappels Caml.

- La fonction `premier` correspond au testeur « premier » sur les séquences.
- `&&` et `||` correspondent respectivement à « et puis » et « ou alors ».
- `[]` et `::` dénotent respectivement les constructeurs « [] » (séquence vide) et « o » (ajout à gauche d'une séquence).
- `abV` et `abNV` dénotent respectivement les constructeurs « \wedge » (arbre binaire vide) et « $/_,_,_ \backslash$ » (arbre binaire non vide), correspondant au type Caml `ArBin`. Les sélecteurs sont `Gauche`, `Droit` et `Racine`.

1 Lecture d'expressions Caml

Q1. On soumet chacune des expressions suivantes à l'interpréteur Caml. Quand l'expression est correcte, donner sa valeur et son type. Quand l'expression est incorrecte, préciser l'origine de l'erreur. Par exemple, pour préciser l'erreur concernant l'expression `'9'+12;;` on indique que `9` est de type `char`, alors que l'opérateur Caml `+` est spécifiée avec deux paramètres de type `int`.

- a) `2.5 /. 1.0;;`
- b) `2/3;;`
- c) `(3>4) && (3<=(if 3>4 then 4 else 3)) || false;;`
- d) `(2+3,"2+3");;`
- e) `if 1 !=1 then 2;;`
- f) `let x = 1 in
 x + (let x = 2*x in
 2*x);;`
- g) `[1]::[[1]];;`
- h) `premier(["123","456"]);;`

2 Somme des feuilles

On veut réaliser en Caml une fonction calculant la somme des feuilles d'un arbre binaire d'entiers.

2.1 Modèle 1

La spécification de la fonction `SommeFeuille` est la suivante :

`SommeFeuille` : un arbre binaire d'entiers \rightarrow un entier
 { Par exemple, soit $unArbre = //\wedge, 1, \wedge\backslash, 2, //\wedge, 3, \wedge\backslash, 4, \wedge\backslash\backslash$ dans $SommeFeuille(unArbre) = 4$ }

Q2. Dessiner l'arbre correspondant à $unArbre$.

Q3. Donner une expression Caml correspondant à $unArbre$.

Q4. Compléter la réalisation récursive en Caml ci-dessous de la fonction `SommeFeuille` :

```
let rec (SommeFeuille : ...) = fonction
  a -> if EstVide_ab(a) then ....
      else ....
;;
```

Q5. Donner une autre réalisation récursive de la fonction `SommeFeuille` en Caml n'utilisant pas d'expression conditionnelle (c'est-à-dire sans `if _ then _ else`).

2.2 Modèle 2

On s'intéresse maintenant aux arbres non vides, et à la fonction `SommeFeuilleNV` dont la spécification est :

`SommeFeuilleNV` : un arbre binaire non vide d'entiers \rightarrow un entier.
 { Par exemple, $SommeFeuilleNV(unArbre) = 4$ }

Q6. Donnez une réalisation en Caml de la fonction `SommeFeuilleNV`.

Une trace de l'exécution de la fonction `SommeFeuilleNV` est partiellement reproduite ci-dessous, les parties manquantes étant symbolisées par des `....`.

NB : un groupe de `....` peut correspondre à plusieurs lignes.

```
#trace("SommeFeuilleNV") ;;
La fonction SommeFeuilleNV est dorénavant tracée.

#SommeFeuilleNV(unArbre) ;;
SommeFeuilleNV <-- abNV
                (abNV (abV, 1, abV), 2,
                 ....)
SommeFeuilleNV <-- ....
SommeFeuilleNV <-- abNV (abV, ...., abV)
SommeFeuilleNV --> 3
....
- : int = 4
```

- Q7.** Indenter et compléter cette trace, afin de faire apparaître la structure des appels récursifs. Indiquer, par exemple par des couleurs, la correspondance entre un appel récursif et le résultat de cet appel.
- Q8.** Combien d'appels récursifs à la fonction `SommeFeuilleNV` sont engendrés sur cet exemple?
- Q9.** Généraliser la réponse à la question précédente pour un arbre non vide d'entiers quelconque de n noeuds. Justifier.

3 Gestion d'un historique

On considère des événements historiques. La description d'un événement historique comporte :

- Un nom représenté par une séquence non vide de caractères. Le nom caractérise l'évènement : deux évènements différents ont des noms différents.
- La date à laquelle l'évènement a eu lieu. La date est représentée sous forme d'un triplet d'entiers correspondant au jour, au mois et à l'année.

On appelle historique un ensemble d'évènements. Un historique sera représenté sous forme d'une séquence ordonnée d'évènements. L'ordre utilisé est l'ordre chronologique. Lorsque deux (ou plus) évènements ont lieu à la même date, aucun ordre particulier n'est imposé entre eux dans la séquence. Par exemple, voici un historique d'un étudiant de MIAS dans la notation du cours : `histo = [< "debut cours",<1,9,2001> >, < "fin cours",<19,04,2002> >, < "exam tp",<26,04,2002> >, < "vacances",<1,7,2002> >, < "plage",<1,7,2002> >, < "jour de l'an",<1,1,2003> >]`. Afin de modéliser ces informations en Caml, on introduit différents types.

- Q10.** Compléter les définitions de ces types ci-dessous :

```

type jour == int (* restreint à 1..31 *) ;;
type mois == int (* restreint à 1..12 *) ;;
type année == int (* > 0 *) ;;
type date == jour * mois * année ;;
type nomEv == .... ;;
type ev == .... ;;
type historique == .... ;;

```

Étant donné un historique, la fonction `DateEv` permet d'obtenir la date à laquelle a eu lieu un évènement de nom donné :

`DateEv` : un historique, un `nomEv` \rightarrow un booléen, une date

{ Soit $\langle ok, d \rangle = \text{DateEv}(h, n)$. S'il existe un évènement de nom n dans l'historique h , alors d est la date de cet évènement et ok a pour valeur vrai ; sinon, ok a pour valeur faux et d est une date quelconque. }

- Q11.** Compléter la réalisation récursive en Caml ci-dessous de la fonction `DateEv` :

```

let rec (DateEv : ...) = function
  ([], n) -> (false, (1,1,1))
  | ....
;;

```

La fonction `ajoutEv` permet d'ajouter – en respectant l'ordre chronologique – un nouvel évènement à un historique donné :

ajoutEv : un historique, un ev \rightarrow un historique

{ Précondition : l'évènement n'appartient pas déjà à l'historique.

Par exemple, soient *e1* et *e2* les évènements \langle "ds", \langle 16, 3, 2002 \rangle \rangle et \langle "ren-
trée", \langle 1, 9, 2002 \rangle \rangle

- ajoutEv(histo, *e1*) = [\langle "debut cours", \langle 1, 9, 2001 \rangle \rangle , \langle "ds", \langle 16, 3, 2002 \rangle \rangle ,
 \langle "fin cours", \langle 19, 4, 2002 \rangle \rangle , \langle "exam tp", \langle 26, 4, 2002 \rangle \rangle , \langle "vacances", \langle 1, 7, 2002 \rangle \rangle ,
 \langle "plage", \langle 1, 7, 2002 \rangle \rangle , \langle "jour de l'an", \langle 1, 1, 2003 \rangle \rangle]

- ajoutEv(histo, *e2*) = [\langle "debut cours", \langle 1, 9, 2001 \rangle \rangle , \langle "fin cours", \langle 19, 4, 2002 \rangle \rangle ,
 \langle "exam tp", \langle 26, 4, 2002 \rangle \rangle , \langle "vacances", \langle 1, 7, 2002 \rangle \rangle , \langle "plage", \langle 1, 7, 2002 \rangle \rangle ,
 \langle "rentrée", \langle 1, 9, 2002 \rangle \rangle , \langle "jour de l'an", \langle 1, 1, 2003 \rangle \rangle]

}

Q12. Donner une réalisation récursive en Caml de la fonction ajoutEv. On pourra éventuellement introduire une fonction intermédiaire; dans ce cas, on donnera sa réalisation.

La fonction LesNomsDeLannee permet d'extraire tous les noms d'évènements ayant eu lieu une année donnée parmi un historique, sous la forme d'une séquence (éventuellement vide) :

LesNomsDeLannee : un historique, une année \rightarrow une séquence de nomEv

{ Par exemple :

- LesNomsDeLannee(histo, 2001) = ["debut cours"]

- LesNomsDeLannee(histo, 2002) = ["fin cours", "exam tp", "vacances", "plage"]

}

Q13. Donner une réalisation récursive en Caml de la fonction LesNomsDeLannee. On appréciera les solutions exploitant l'ordre chronologique de l'historique.