Université de Marne-la-Vallée
Licence d'informatique et IUP GMI
Examen de programmation fonctionnelle du 22 janvier 2001
Durée 2 heures. Tous les documents sont autorisés.

Tous les exercices sont indépendants. N'oubliez pas de tourner la page.

Exercice 1.— (4 points)
On définit en Caml le type suivant `'a arbre` qui permet de manipuler des arbres binaires dont les étiquettes des noeuds sont formés d'éléments de type `'a`.

```
type 'a arbre =
    Vide
  | Noeud of 'a * 'a arbre * 'a arbre ;;
```

a) (2 points)
Écrire une fonction Caml `bord` qui calcule la suite des étiquettes des feuilles d'un arbre binaire obtenues dans l'ordre préfixe (ou encore de gauche à droite). Une feuille est un nœud dont les deux sous-arbres gauche et droit sont `Vide`.

b) (2 points)
Écrire une version avec accumulateur pour la question précédente.

Exercice 2.— (10 points)

a) (2 points)
Écrire une fonction `loop` qui prend en argument un prédicat `p`, une fonction `f` et un élément `x` auquel peuvent s'appliquer la fonction et le prédicat et qui rend la première valeur de $f^{(n)}(x)$, pour n entier positif ou nul, qui satisfait `p`. Un prédicat en Caml est fonction de type `'a-> bool`. Quel est le type de `loop` ?

b) (2 points)
Écrire une fonction `exists` qui prend en argument un prédicat `p` et un liste d'éléments auxquels s'appliquer `p` et qui teste s'il y a au moins un élément qui satisfait le prédicat `p` dans la liste. Quel est le type de `exists` ?

c) (2 points)
Écrire une fonction `find` qui trouve le premier élément qui satisfait le prédicat passé en premier argument dans la liste passée en deuxième argument. Quel est le type de `find` ?

On veut écrire des fonctions Caml pour manipuler des relations binaires. Une relation binaire R sur un ensemble E est définie comme une partie de $E \times E$. Les éléments de R sont donc des couples (x, y) avec x, y dans E. On représente une telle relation par une fonction, notée encore R, de E dans l'ensemble des parties de E. Cette fonction associe à tout x de E l'ensemble des éléments en relation avec y :

$$R(x) = \{y \mid (x, y) \in R\}.$$

On étend R en une fonction définie sur les parties de E en posant :

$$R(X) = \bigcup_{x \in X} R(x),$$

pour toute partie X de E. Pour une relation R présentée de cette façon, la fermeture réflexive et transitive R^* de R est définie par :

$$R^*(x) = \bigcup_{n=0}^{\infty} R^n(x),$$

avec

$$R^0(x) = \{x\}, \quad \text{et } R^{n+1}(x) = R(R^n(x)).$$

Pour la représentation en Caml, les ensembles seront représentés par des listes. Pour simplifier, on ne fera pas l'hypothèse que les listes représentant des ensembles sont toujours non redondantes. Dans une liste représentant un ensemble, un élément de l'ensemble peut donc apparaître plusieurs fois. Une relation R sur un ensemble E sera représentée en Caml par une fonction `rel : E -> E list`. On supposera que pour toute valeur x de E, `rel x` est la liste éventuellement redondante des éléments de $R(x)$.

d) (2 points)
Écrire une fonction `flat_map` telle que pour toute fonction Caml `rel` représentant une relation R, `flat_map rel` représente R étendue aux parties de E. Quel est le type de `flat_map` ?

Étant données une relation R sur un ensemble E et une propriété *Prop* des éléments de E, on s'intéresse au problème suivant : soit un élément s de E. Trouver un élément t de E tel que $t \in R^*(s)$ et *Prop(t)*. Une proposition *Prop* sur un ensemble E est représentée en Caml par une fonction `prop : E -> bool`. On veut résoudre le problème énoncé ci-dessus à l'aide d'une fonction Caml unique `solve` susceptible de s'appliquer à toutes les propriétés souhaitées. Cette fonction aura pour type :
`('a -> 'a list) -> ('a -> bool) -> 'a -> 'a`.
Une façon naïve de résoudre le problème consiste à engendrer successivement tous les ensembles $R^n(s)$ jusqu'à ce que l'un d'eux contienne un élément satisfaisant la propriété cherchée.

e) (2 points)
Écrire une fonction `solve` implantant cette stratégie et utilisant les fonctions `loop`, `flat_map`, `exists` et `find`.

Exercice 3.— (4 points)
Indiquer si les formules suivantes sont des tautologies, des antilogies ou ni l'un ni l'autre. On fera une preuve sémantique.

a) (2 points)
$\vdash (p \to (q \vee r)) \to ((p \wedge \neg q) \to r)$;
b) (2 points)
$\vdash (p \to ((q \to r) \to p)) \to (p \to r)$.

Exercice 4.— (4 points)
Si p, q, r désignent des atomes (ou variables) du calcul des propositions, montrer que :
a) (2 points)
$\vdash (p \to (q \vee r)) \to ((p \wedge \neg q) \to r)$;
b) (1 point)
$(p \to (q \to r)) \to q \to p \to r$
Le système de déduction à utiliser est le système de Hilbert présenté en cours. On pourra utiliser tous les résultats du cours (notamment le théorème de déduction), à l'exclusion toutefois du théorème de complétude.
c) (1 point)
Définir une fonction Caml du type :
```
('a -> 'b -> 'c) -> 'b -> 'a -> 'c
```