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

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

Exercice 1.— (10 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) Ecrire une fonction `renverse` qui prend an argument un arbre dont les étiquettes sont des couples et renvoie l'arbre obtenu à partir de celui-ci en renversant tous les couples. Le type de `renverse` sera : `('a * 'b) arbre -> ('b * 'a) arbre`

b) Ecrire une fonctionnelle `tree_map` de type `('a -> 'b) -> ('a arbre) -> ('b arbre)` qui prend en argument une fonction `f` de type `'a -> 'b` et un arbre `t` de type `'a arbre` et renvoie un arbre de type `'b arbre` obtenu en appliquant `f` sur chaque étiquette d'un noeud de `t`.

c) Réécrire `renverse` en utilisant `tree_map`.

d) Écrire une fonction booléenne `test` qui prend en argument deux arbres de type `'a arbre` est renvoie vraie si et seulement si les deux arbres ont la même forme, c'est-à-dire s'ils sont identiques à un changement près des étiquettes.

e) Écrire une fonction `find` qui prend en argument un arbre de type `'a arbre` et un prédicat `p` qui peut s'appliquer aux éléments de l'arbre et qui trouve le premier élément de l'arbre passé en argument, en suivant une exploration préfixe, qui satisfait le prédicat passé en deuxième argument. On lèvera une exception lorsqu'aucun élément de l'arbre ne satisfait le prédicat. Quel est le type de `find` ?

Exercice 2.— (4 points)
On considère un graphe orienté `g` qui est donné sous forme de listes de couples d'entiers représentant les arêtes. Le graphe ci-dessous par exemple a trois flèches $1 \to 2$, $1 \to 3$ et $2 \to 3$.

```
let g=[(1,2);(1,3);(2,3)];;
```

Écrire, à l'aide de fonctions intermédiaires que vous définirez et écrirez vous-même, une fonction Caml `chemin` qui prend en argument deux entiers `s` et `t` représentant deux sommets, un graphe `g`, et renvoie vraie ss'il existe un chemin de `s` à `t`. Ainsi sur l'exemple, on obtient

```
chemin 1 3 g;;
- : bool = true
```

Exercice 3.— (2 points)
Effectuer une résolution restreinte et ordonnée de la famille de clauses suivantes :
$C = \{p \vee r, q \vee r, q \vee s, \neg p \vee \neg q, \neg r \vee \neg s, \neg q \vee \neg r\}$.
Que peut-on en déduire sur la famille C ?

Exercice 4.— (4 points)
Si p, q, r désignent des atomes (ou variables) du calcul des propositions, montrer que :

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

b) (1 point)
$\vdash (p \to q \to r) \to p \to q \to q \to r$.

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 -> 'a) ->'a -> 'b -> 'b -> 'a`