UNIVERSITÉ D'ORLÉANS — DEUG MIAS

Informatique — Année **2001-2002**

Feuille d'exercices OCaml n°6 - CAML : polymorphisme et listes

---

Exercice 1. Définir `paires` de type `'a list -> ('a * 'a) list` qui retourne la liste de toutes les paires d'éléments de son argument. Attention au cas de la liste vide.

Exercice 2.
Le tri rapide (*Quick Sort*) d'une liste `l=(x::l')` s'effectue de la manière suivante : la liste `l'` est décomposée en deux sous-listes `l_{inf}` et `l_{sup}` telles que `l_{inf}` est la liste des éléments de `l'` inférieurs ou égaux à `x` et `l_{sup}` la liste des éléments de `l'` supérieurs à `x`. Si `T_{inf}` (resp. `T_{sup}`) est la liste `l_{inf}` (resp. `l_{sup}`) triée, alors la liste `l` triée est obtenu en concaténant les listes `T_{inf}`, `[x]` et `T_{Sup}`.

Écrire une fonction qui trie, en utilisant le *Quick sort*, les éléments d'une liste suivant une fonction de comparaison donnée en paramètre. Typer cette fonction. On définira d'abord la fonction `partition\_inf` (resp. `partition\_sup`) qui prend en arguments une liste et un élément `elem` et qui renvoie la liste formée des éléments de la liste inférieurs ou égaux (resp. supérieurs) à `elem`.

Etant donné une relation d'ordre sur un type, en déduire une fonction qui trie par ordre lexicographique des couples d'objets ce type.

Exercice 3. Arbres binaires feuilles.

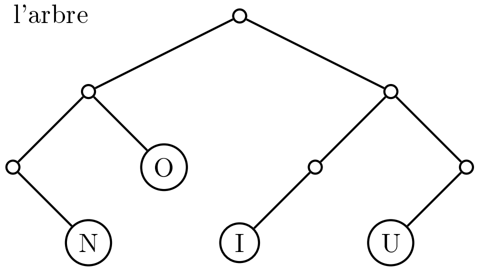
On rappelle qu'un *arbre binaire feuille* (ABF) est un arbre binaire dont seules les feuilles contiennent des informations.

1. Définir un type polymorphe ABF (ie dont le type des informations aux feuilles n'est pas encore déterminé).
2. Une application des ABF est le codage/décodage. Etant donnés un ABF et un code représenté par une liste de booléen, on obtient l'information codée en parcourant en même temps l'arbre et la liste : si la tête de liste vaut `true`, on va à gauche dans l'arbre, sinon on va à droite. Si l'arbre et la liste sont bien formés, on arrive sur une feuille lorsque la liste est vide : l'information cherchée s'y trouve.

   (a) Ecrire une fonction `decode_info` qui prend un ABF et un code et retourne l'information correspondante ou une erreur.

   (b) Ecrire une fonction `code_info` qui prend un ABF et une information et retourne le code correspondant ou une erreur.

   (c) Appliquer les fonctions précédentes aux ABF contenant des lettres. Par exemple, étant donnée l'arbre

   La liste `[true ; true ; false]` code N, I est codé par `[false ; true ; true]`, `[true ; true ; true]` est un mauvais code, et A n'en a pas.

   (d) Comment coder les mots ? Déterminer les types et fonctions pour coder et décoder des mots (par exemple « oui » et « non » dans l'arbre ci-dessus).

Exercice 4. Dites ce que calcule la fonction suivante :

```
let rec ff = function
                  [ ]    -> 1
                | x::l   -> 2 * ff l ;;
```