Université Paris 7 - DEUG-MIAS Année 2004–2005

TD de *Logique et Circuits* n° 2

Introduction à Caml (2)

Exercice 1 On veut définir un type `carte` qui représente un jeu de 32 cartes (quatre couleurs et des valeurs égales à 7, 8, 9, 10, Valet, Dame, Roi, As). Définir un type `couleur` dont les valeurs possibles sont Trèfle, Cœur, Carreau et Pique.

Définir le type `carte` et écrire une fonction de construction pour ce type qui, à un couple `int * couleur` associe une carte (11 → Valet, 12 → Dame, 13 → Roi, 1 → As) et retourne une erreur s'il n'y a pas de valeur correspondante.

On suppose qu'on joue à la belote. À ce jeu, une des couleurs est privilégiée (c'est l'« atout ») et la valeur des cartes de cette couleur est différente :

	As	Roi	Dame	Valet	10	9	8,7
Couleur atout	11	4	3	20	10	14	0
Couleur non atout	11	4	3	2	10	0	0

Écrire une fonction qui, étant données une carte et la couleur de l'atout, renvoie la valeur de la carte.

Exercice 2 On considère les expressions suivantes, extraites d'un programme qui peut comporter des définitions antérieures.

```
type nombre = Ent of int | Dec of float;;

let x =
  if y = f z (y = h z) then
    match (g y) with
    | z, true -> Dec y
    | _ -> Ent z
  else Dec (h z);;
```

Déterminer les types des différents identifiants qui apparaissent dans la dernière expression.

Exercice 3 On définit la fonction `applique` par :

```
let applique f a = f a;;
```

Quel est le type de `applique` ? de `applique applique` ?

Exercice 4 Écrire une fonction `compose` qui, à partir d'une suite de deux fonctions `x` et `y` calcule la fonction composée `z` (définie pour tout t par $\mathtt{z}(t) = \mathtt{x}(\mathtt{y}(t))$. Quel est le type de `compose` ?

Exercice 5 On utilise la fonction `compose` définie dans l'exercice précédent et on définit le type `fct` et (de manière incomplète) la fonction `f` :

```
type ('a, 'b) fct = Dec of (float -> 'a) | Ent of (int -> 'b);;

let f g h =
  match (g,h) with
  | Dec a, Ent b -> ... (compose a b)
  | Dec a, Dec b -> ... (compose b a)
  | Ent a, Dec b -> ... (compose a b)
  | Ent a, Ent b -> ... (compose ... ...);;
```

Compléter la définition de `f` et donner son type.

Exercice 6 On définit un type énuméré `jour` dont les valeurs sont : `Lundi`, `Mardi`,... et un type énuméré `mois` dont les valeurs sont aussi des constantes : `Janvier`,...
Écrire la fonction `jsuivant` qui calcule le jour suivant. On supposera qu'il existe aussi une fonction `msuivant` pour les mois. Écrire une fonction qui, étant donnés une suite de deux paramètres `mois` et `int` (représentant l'année) rend la longueur du mois (on supposera, pour simplifier,que les années multiples de 4 sont bissextiles).
Définir un type abstrait `date` tel que, pour chaque élément de type `date`, on puisse accéder aux informations suivantes : le jour de la semaine, le numéro du jour dans le mois, le mois et l'année. Écrire une fonction qui, étant donné une date, retourne la date du lendemain. On pourra écrire cette fonction de manière concrète ou abstraite.

Exercice 7 Une entreprise de transports publics dispose de trois types de tickets :

- les `coupons` qui ont une durée (en nombre de mois, maximum un an) et un nombre de zones d'utilisation fixés (représenté par un entier $n \in [1;8]$, ce qui signifie que le coupon est valable de la zone 1 à la zone n),
- les `cartes` qui sont utilisables un nombre fixé de fois dans une période d'un an, mais pour n'importe quel trajet,
- les tickets `simples` qui ne servent qu'à un trajet.

1. Définir le type abstrait `ticket`.
2. Le prix d'un ticket (arrondi à l'euro inférieur) est donné par les formules suivantes :
   - coupon : $20(m+1)z$ (m :nbe de mois, z :nbe de zones),
   - carte : $5 + t/2$,
   - simple : 1.

   Écrire la fonction `prix` en utilisant les opérations abstraites dont on dispose sur les tickets.
3. Écrire une fonction `choix` qui prend comme suite de paramètres la durée d'utilisation, la zone et le nombre de trajets à effectuer, et retourne le ticket le moins cher (s'il est moins cher d'acheter des tickets simples, on retourne un seul ticket simple).

Exercice 8 On définit les types suivants :

```
type place = Pl of int * int;;
type deplacement = Dep of (place->place);;
```

On considère un point sur un écran. Une `place` est caractérisée par ses coordonnées. Un déplacement est une opération qui permet d'aller d'une place à une autre en faisant certain nombre de pas horizontaux et un certain nombre de pas verticaux. Ainsi, `d x` est la place atteinte à partir de la place `x` avec le déplacement `Dep d`.
Écrire une fonction qui, à tout couple de places, associe le déplacement qui permet d'aller de la première à la seconde.
Écrire une fonction qui prend comme argument un déplacement et calcule le déplacement inverse.