Institut National Universitaire Champollion

# TP 3 - Fonctions définies par filtrage

**Programmation fonctionnelle en Caml**
**L3 INFO - Semestre 6**

## 1. Première fonction définie par filtrage

Écrire une fonction `entier : int -> string`, définie par un filtrage exhaustif, qui à un entier $n$ associe la chaîne de caractère "zero" si $n$ vaut 0, "un" si $n$ vaut 1 et dans les autres cas, "pair" si $n$ est pair, et "impair" si $n$ est impair.

```
entier(0) ; ;
-  : string = "zero"
entier(3) ; ;
-  : string = "impair"
```

## 2. Un peu de géométrie

Nous souhaitons distinguer dans le plan réel : l'origine $(0,0)$, les points de l'axe des abscisses, les points de l'axe des ordonnées, et les autres que nous partageons entre les points du demi-plan $x > 0$ et les points du demi-plan $x < 0$.
Écrire une fonction `point : float * float -> string`, obligatoirement définie par filtrage exhaustif, qui à un couple de réels représentant un point du plan associe sa catégorie.

```
#point(0.,0.) ; ;
-  : string = "Origine"
#point(0.,3.) ; ;
-  : string = "Axe des ordonnées"
#point(2.,-3.) ; ;
-  : string = "point du demi plan x>0"
```

## 3. Opérations

Écrire une fonction `operation : int * int * char -> int`, obligatoirement définie par un filtrage exhaustif, qui à deux entiers $x$ et $y$ et à un caractère $c$ associe :

- Si $c$ est un des caractères $+, -, *$ ou $/$ : le résultat de l'opération correspondante entre $x$ et $y$ (On veillera à ne pas faire de division par 0)
- Un message d'erreur sinon.

```
#operation(7,3,'+') ; ;
-  : int = 10
#operation(7,3,'-') ; ;
-  : int = 4
#operation(7,0,'/') ; ;
```

```
Exception non rattrapée : Failure "division par zéro !"
#operation(7,0,' !') ; ;
Exception non rattrapée : Failure "Ce n'est pas une opération connue."
```

## 4. Calcul de TVA

La TVA (taxe sur la valeur ajoutée) est une taxe qui s'ajoute à la valeur des marchandises. Elle s'exprime comme un pourcentage du prix de base (appelé prix hors taxe), pourcentage qui dépend de la nature des marchandises et qui est appelé taux de TVA . Les prix TTC (toute taxe comprise) est le prix hors taxe + le montant de la TVA.
Par exemple un produit à 200€(hors taxe) assujetti à une TVA à 12% aura une TVA de 24€et un prix TTC de 224€.

1. Écrire une fonction `prixTTC : float * float -> float` calculant le prix TTC d'une marchandise en fonction de son prix hors taxe et du taux de la TVA.

   ```
   #prixTTC(200.,12.) ; ;
   - : float = 224.0
   ```

2. Écrire une fonction `prix : string -> float * float`, définie par filtrage, qui à un article de la liste suivante associe son prix unitaire hors taxe et le montant de la TVA :

   | **Article** | **Prix unitaire** | **Taux TVA** |
   |---|---|---|
   | pain | 1,05 | 5,5 % |
   | conserve | 3,50 | 7 % |
   | disque | 12,30 | 18,6 % |
   | bijou | 356 | 33 % |

   Pour tout autre article, on affichera le message d'erreur "Article indisponible".

3. Écrire une fonction `sommeAPayer : string * int -> float`, utilisant les fonctions précédentes, qui à un nom d'article et au nombre d'articles achetés associe la somme TTC à payer, arrondie à la dizaine de centimes la plus proche (utilisez la fonction `arrondi` du TP 2).

## 5. Un calendrier perpétuel

Le but de ce problème est d'écrire une fonction qui à une date donnée sous la forme de trois entiers (jour, mois, année) associe le jour de la semaine correspondant à cette date.

1. Écrire une fonction `formule : int * int * int * int -> int` qui à un quadruplet d'entiers $(j,m,p,s)$ associe l'entier

   $$j+(48m-1)/5+p/4+p+s/4-2*s$$

2. Écrire une fonction `decoupe : int -> int * int`, qui à un entier $n$ associe son quotient et son reste dans la division entière par 100.

   ```
   decoupe(2014) ; ;
   - : int * int = 20, 14
   ```

3. Écrire une fonction `deuxMoisAvant : int * int -> int * int`, définie par filtrage, qui étant donné un couple (mois, année) associe le couple correspondant à la date antérieure de 2 mois à la date donnée.

```
deuxMoisAvant(2,2014) ; ;
-  : int * int = 12, 2013
```

4. Écrire une fonction `leJour : int -> string`, définie par filtrage non exhaustif qui à 0 associe "Dimanche", 1 associe "Lundi"... et à 6 associe "Samedi".

```
leJour(2) ; ;
-  : string = "Mardi"
```

5. Écrire une fonction `modulo7 : int -> int` qui étant donné un entier donné un entier $n$ renvoie le reste de la division euclidienne de $n$ par 7.
   Attention ! En Caml, $12\ mod\ 7$ donne 5 mais $-12\ mod\ 7$ donne $-5$. Donc `modulo7(n)` est égal à $nmod7$ s'il est positif ou nul et $(n\ mod\ 7)+7$ sinon.

```
modulo7(20) ; ;
-  : int = 6
modulo7(-4) ; ;
-  : int = 3
```

6. Il est possible de retrouver le jour de la semaine à partir d'une date donnée par la méthode suivante :
   - Partant de la date (jour, mois, année), on calcule $(M,A)$, le mois et l'année précédent de deux mois la date (mois, année).
     Par exemple, si la date est le $(14,10,2014)$, on obtient $(M,A)=(8,2014)$.
   - On sépare les 4 chiffres de l'année $A$ en un couple $(S,P)$. $S$ correspond aux deux premiers chiffres de $A$ et $P$ à ses deux derniers chiffres.
     Sur notre exemple, on obtient $(S,P)=(20,14)$.
   - On calcule alors un entier $K$ par la formule :

$$K = jour + (48 * M - 1)/5 + P/4 + P + S/4 - 2 * S$$

     On reconnaît la fonction `formule` et on obtient sur notre exemple $K=72$.
   - Il reste à faire la division euclidienne de $K$ par 7 et à prendre le jour correspondant de la semaine.
     On obtient ici $72 = 7*10+2$ et le 2ème jour est Mardi.

   Écrire, en utilisant les fonctions précédentes, une fonction `quelJour : int * int * int -> string` mettant en œuvre cette méthode pour calculer le jour correspondant à une date donnée.

```
quelJour(14,7,1789) ; ;
-  : string = "Mardi"
```