# - TD2 OCAML -

## Exercice sur le `match`

1. Faites une fonction qui convertit les booléens en entier $0 = false, 1 = true$
2. Faites la fonction de conversion inverse
3. Faites une fonction `int2roman` qui transforme les chiffres en chiffre Romain.

## Récursivité

1. Faites la fonction factoriel ($n! = \prod_{i,1,n} i$) par deux manières: en utilisant le if et le match
2. Faites la fonction permettant de calculer la série $1 + 1/2 + 1/4 + 1/8 + 1/16, \cdots$ Testez la. Vers quelle valeur converge-t-elle ? (hint: `float_of_int` convertit un entier en réel.)
3. Faites la fonction permettant de calculer la suite $0, 1, 1, 2, 3, 5, 8, 13, \cdots$. Cette suite est connue sous le nom de *suite de Fibonacci*. ( `http://fr.wikipedia.org/wiki/Suite_de_Fibonacci`)
4. le nombre de combinaison $C_n^p = \frac{n!}{(n-p)!p!}$ peut être calculé récursivement en s'appuyant sur la propriété suivante

$$\begin{cases} C_n^0 &= 1 \\ C_n^n &= 1 \\ C_n^p &= C_{n-1}^{p-1} + C_{n-1}^p \end{cases}$$

   Faites une fonction récursive permettant de calculer la combinaison $C_n^p$.
5. Faites une fonction permettant de calculer une fonction d'Ackermann définie mathématiquement comme suit :

$$\begin{cases} a(0,n) &= n+1 \\ a(m,0) &= a(m-1,1) \\ a(m,n) &= a(m-1,a(m,n-1)) \end{cases}$$

   `http://fr.wikipedia.org/wiki/Fonction_d'Ackermann`

## Récursivité et liste

Faites les fonctions suivantes et testez les :

1. `after0` : retourne la liste située après le premier 0. (`after0 [1;0;2;0;3]` → `[2;0;3]`)
2. `before0` : retourne la liste située avant le premier 0. (`before0 [1;0;2;0;3]` → `[1]`)
3. `remove0` : ôte tous les 0 d'une liste.
4. `add0` : ajoute un 0 un élément sur 2 (`add0 [1;2;3]` → `[1;0;2;0;3]`).
5. `opposite` : remplace toutes les valeurs ($a$) de la liste par leur opposé ($-a$).
6. Faites une fonction `mem l a` qui recherche si $a$ fait partie de la liste $l$

7. Faites la même fonction en considérant que $l$ est triée par ordre croissant.

8. Faites la fonction `nth i l` qui ressort le $i^{eme}$ élément d'une liste $l$ en considérant que l'élément en tête de liste est à la position 1.

9. Faites la fonction `append l1 l2` qui "colle" la liste après la liste $l_1$ (En OCAML il existe aussi un opérateur permettant de faire cela. il s'agit de `@`.)

## Exercices supplémentaires

1. Faites une fonction `flatten` qui applatit une liste de listes en une liste normale de valeurs. Par exemple `applatit [ [1] ; [2;3] ;[4;5;6] ]` retourne `[1;2;3;4;5;6]`

2. Faites une fonction `clean` qui retire valeurs répétées plusieurs fois dans une liste en ne conservant que la première occurrence de la valeur. `clean [1;1;2;1;1;2;2;3;4;3]` retourne `[1;2;3;4]`