# Expressions, Fonctions, Récursivité

Pour faire gagner du temps, un fichier "squelette" est disponible sur le net. Penser à le récupérer.

**Exercice 1 : Prise en main**

Parmi les expressions suivantes, lesquelles sont correctes en Lisp ? Quelles sont leurs valeurs ?
- `- 2 1`
- `(+ (*) (+))`
- `(* (+ 2 5) ((* 3 4) (- 1 2)))`
- `(12 + (* 3 4))`
- `(+ 2 (* 5 (log (8))))`
- `(+ 1 2 3 4 5 6)`
- `(log -2)`
- `(defun 3 (+ 2 1))`

**Exercice 2 : Prise en main (suite)**

Définir les constantes a, b, c, d, e ci-dessous en utilisant alternativement la forme spéciale "defun" et la forme spéciale "defparameter" :
- $a = \sqrt{1 + \sqrt{2 + \sqrt{3}}}$
- $b = (2 + 3).(4 + 5 + 6)$
- $c = \log\left(99^2 + 3\right)$
- $d = \frac{a+b}{a-b}$
- $e = \frac{a+b}{a+2b} - \sqrt{\frac{a+2b}{a+b}}$

**Exercice 3 : Trinôme - Utilisation de *cond* / *if***

Remarque sur les fonctions :
Il est d'usage en Lisp de documenter ses fonctions. Pour cela, on intercale une chaîne de caractères descriptive entre les arguments et le corps de la fonction.
Exemple :

```
(defun factorial (n)
  `` Calcule la factorielle de l'entier n.''
  (if (<= n 1)
    1
    (* n (factorial (- n 1)))))
```

Ensuite il est possible de récupérer la documentation d'une fonction grâce au symbole "documentation".
Exemple :

```
(documentation 'factorial 'function)
-> " Calcule la factorielle de l'entier n."
```

Essayer aussi les symboles "apropos" et "inspect"...

- Ecrire une fonction `discriminant` qui, étant donnés 3 arguments $a$, $b$ et $c$ représentant les coefficients d'un trinôme, calcule le discrimant de ce trinôme
- Ecrire deux fonctions `racine1` et `racine2` qui, étant donnés 3 arguments $a$, $b$ et $c$ représentant les coefficients d'un trinôme, calculent respectivement la valeur de la première racine et de la deuxième la racine du trinôme. Au cas où il n'y a qu'une racine, les deux fonctions renvoient la même valeur

– Ecrire une fonction `carac-racines-trinome` qui, étant donné 3 arguments $a$, $b$ et $c$ représentant les coefficients du trinôme, indique si le trinôme aura 2 racines réelles, 1 racine réelle ou 2 racines complexes. Faire une version avec des "if" puis une version avec "cond".

*Exemple d'utilisation :*

```
* (carac-racines-trinome 1 2 -3)

 ``2 racines réelles''

* (carac-racines-trinome 3 4 5)

 ``2 racines complexes''

* (carac-racines-trinome 1 -2 1)

``1 racine réelle''
```

**Exercice 4 : Quelques fonctions**

Définir les fonctions suivantes : `carre` (d'un nombre), `moyenne` (de deux nombres), `valeur-trinome` (pour un trinôme $ax^2 + bx + c$ donné par ses coefficients $a$, $b$ et $c$).

**Exercice 5 : Pair / Impair**

Définir deux fonctions `pair` et `impair` qui déterminent la parité de leur argument suivant le principe : 0 est pair, et $n$ est pair (respectivement, impair) si $n - 1$ est impair (respectivement, pair).

**Exercice 6 : Pgcd**

Ecrire une fonction `pgcd` donnant le plus grand diviseur commun de ses deux arguments, en se basant sur l'algorithme d'Euclide : le PGCD de $a$ et $b$ est égal au PGCD de $b$ et de ($a$ mod $b$), pourvu que $b > 0$.

**Exercice 7 : Utilisation de *trace*** (sur brahmane)

*trace* est une fonction très utile permettant de visualiser tous les appels à une fonction dont le nom est passé en paramètre. Elle est particulièrement intéressante lorsqu'il s'agit de visualiser le déroulement des appels dans une fonction récursive. Dans la suite de cette feuille, essayez là sur les différentes fonctions que vous aurez écrites. Par exemple, faites (*trace* `pair`), (*trace* `impair`). Puis (`pair` 10). (Pour retirer le mode trace sur une fonction, (*untrace* nom-fonction)).
**NB :** à l'heure actuelle, la fonction *trace* ne fonctionne pas correctement sur les machines des salles de TD. Pour l'utiliser, il faut se loguer sur *brahmane*.