Programmation fonctionnelle avec OCaml
TP 2

Objectifs :
- Fonctions récursives : récursivité directe, récursivité terminale.
- Factorisation et abstraction : itérateur, ordre supérieur (passage de fonction en paramètre), polymorphisme, généricité.

Aggrégation

Dans un nouveau fichier, on va écrire la fonction `sigma` (et des généralisations) qui correspond au symbole Σ dans une sommation :

$$\sum_{i=a}^{b} f(i)$$

Pour chaque généralisation, on testera la cohérence des résultats à l'aide du même exemple, tel que la somme des premiers carrés : $\sum_{i=1}^{10} i^2 = 385$.

1. On suppose que a et b sont des entiers ; $\sum_{i=a}^{b} f(i)$ signifie alors $f(a)+f(a+1)+\ldots+f(b)$ si $a \leq b$ et 0 sinon. Écrire cette première version de la fonction `sigma` avec trois arguments (f, a et b) dans le style **récursif direct** et en n'utilisant que deux cas (arrêt et général).
2. Écrire une fonction équivalente en style **récursif terminal**.
3. Généraliser[1] (votre version préférée) avec un incrément quelconque (i.e. pas nécessairement 1) pris en paramètre supplémentaire.
4. Au lieu d'*ajouter* un incrément quelconque, généraliser en prenant en paramètre une *fonction s* qui calcule l'argument suivant : $f(a) + f(s(a)) + f(s(s(a))) + \cdots + f(b)$
5. Généraliser en prenant en paramètre un prédicat (i.e. une fonction qui renvoie un booléen) quelconque `cont` pour déterminer la borne supérieure des arguments : $f(a) + f(s(a)) + \cdots + f(b)$ où b est le dernier argument tel que `cont b` soit vrai.
6. Généraliser en permettant le remplacement de la somme par n'importe quelle fonction binaire (possédant un élément neutre à prendre également en paramètre).
7. Écrire la fonction factorielle à l'aide de la fonction `sigma`.
8. Écrire une fonction `integrale` qui calcule une approximation de $\int_a^b f(x)dx$ en prenant pour dx un « petit » flottant.
9. Écrire le calcul d'une approximation de π avec une précision `epsilon` donnée en utilisant la série alternée :

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots$$

Indication : utiliser une version récursive terminale pour calculer π avec une précision supérieure à 10^{-5}.

---

1. On pourra vérifier expérimentalement que la conjecture suivante est bien un théorème :

« On oublie toujours de renommer l'appel récursif quand on fait du copier-coller avec des fonctions récursives. »

ainsi que son corollaire :

« Le copier-coller est rarement profitable en programmation. »