TP : récursivité 1
Informatique pour tous

On rappelle qu'une fonction récursive contient généralement :

— Un cas de base, dans lequel la fonction renvoie directement une valeur (l'oubli du cas de base conduit à une fonction qui ne termine pas ! on utilisera Ctrl + I pour arrêter la fonction)

— Un ou des appel(s) récursif(s) sur des sous-problèmes, permettant de résoudre le problème général.

Souvent, une fonction récursive ressemble à ceci :

```
def f(...):
    if ...: # cas de base
        return ...
    return ... f(...) ... # appel(s) récursif(s)
```

I Algorithmes classiques en récursif

1. Écrire une fonction récursive `somme(n)` calculant $\sum_{k=1}^{n} k^2$.
   Vérifier sur des exemples que c'est égal à $\frac{n(n+1)(2n+1)}{6}$.
2. En utilisant la formule de Pascal $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$, écrire une fonction récursive `binom(n, k)` calculant $\binom{n}{k}$.
   (Attention aux cas de bases !)
3. On rappelle que, si $f(a)f(b) \leq 0$, on peut approximer une solution de $f(x) = 0$ à ϵ près sur $[a, b]$ par recherche dichotomique en calculant le milieu $m = \frac{a+b}{2}$:
   (a) Si $b - a \leq \epsilon$ alors m est une approximation qui convient.
   (b) Sinon, remplacer a ou b par m de façon à encore avoir $f(a)f(b) \leq 0$.
   Écrire une fonction récursive effectuant cette recherche dichotomique. Prouver qu'elle termine.

II Algorithme d'Euclide

1. Écrire une fonction récursive `pgcd` renvoyant le PGCD de deux entiers a et b.
   On rappelle que, si $b \neq 0$ et r est le reste de la division euclidienne de a par b, PGCD(a, b) = PGCD(b, r). Vérifier votre fonction.
2. Écrire une fonction récursive `bezout(a, b)` renvoyant trois valeurs (que l'on pourra mettre sous forme de liste ou de 3-uplet) : le PGCD d de a et b et deux entiers u et v tels que $au + bv = d$.
   Indice : appeler récursivement `bezout(b, r)` pour obtenir (d, u', v') tel que $d = bu' + rv'$ puis exprimer u, v en fonction de u', v'.

III Rendu de monnaie

On souhaite savoir combien de façons il y a de rendre n€ avec des pièces de différentes valeurs.
Par exemple, on peut rendre 6€ avec des pièces de 1€, 2€, 3€, 5€ de deux façons : 1+5 et 1+2+3.

1. Écrire une fonction récursive `rendu(n, L)` qui renvoie le nombre de façons de rendre n€ avec des pièces contenus dans la liste `L`.
   Indice : pour rendre n€ avec des pièces p_1, ..., p_k, on peut soit rendre $n - p_k$ avec p_1, ..., p_{k-1}, soit rendre n avec p_1, ..., p_{k-1}.

On veut maintenant afficher tous les rendus possibles.

2. Définir d'abord une fonction `ajouter` telle que, si `LL` est une **liste de listes**, `ajouter(e, LL)` ajoute `e` à chacune des listes de `LL`.
3. En déduire une fonction `rendu2` telle que `rendu2(n, L)` renvoie la liste de tous les rendus possibles pour n€ (chaque rendu étant la liste de ses pièces).
   On pourra utiliser la concaténation des listes avec `+`.
   Indice : pour le cas de base (`L == []`), on renverra `[[]]` (une solution possible : utiliser 0 pièce) si $n == 0$ et `[]` sinon (aucune solution possible).