Travaux Dirigés n°9
DEUG SCIENCES S2
Utilisation de la récursivité

La récursivité est permet la définition de fonctions à l'aide d'elles-même et de conditions initiales. Elle permet parfois l'écriture plus simple de solutions.

---

1. **Factorielle d'un entier**
   La factorielle d'un entier positif est définie par les relations suivantes :
   $$n! = \begin{cases} 1 & \text{si } n = 0, \\ n * (n-1)! & \text{sinon.} \end{cases}$$
   Ecrire une fonction `Factorielle` prenant en argument un paramètre `n` de type `int` et calculant la factorielle de `n`.

2. **Pgcd**
   Ecrire une fonction récursive permettant de déterminer le pgcd de deux entiers passés en paramètre.

3. **Recherche dans un tableau trié**
   Ecrire une fonction de recherche d'un entier dans un tableau rangé en ordre croissant. La recherche se fera récursivement. La fonction reçoit le tableau, les indices de début et de fin de recherche, et renvoie l'indice de l'élement s'il est présent dans le tableau, -1 sinon.

4. **Palindrome**
   Ecrire une fonction récursive permettant de déterminer si une chaine de caractères est un palindrome.

5. **Décomposition d'un entier n en base b**
   Ecrire une procédure récursive recevant deux entiers positifs n et b et affichant la décomposition de l'entier n en base b. On pourra se limiter à $b \leq 10$.

6. **Somme des chiffres d'un entier**
   On veut calculer la somme des chiffres d'un entier n. Par exemple, la somme des chiffres formant 3221 est 8.

   - Trouver un algorithme récursif pour calculer la somme des chiffres formant un entier quelconque.
   - En déduire une fonction `SommeChiffre` prenant en argument un entier `n` et qui implémente l'algorithme précédant.

On considère la décomposition sur la base 10 d'un entier n :

$$n = \sum_{i=0}^{k} 10^i \times n_i.$$

Les critères de divisibilité suivants peuvent se déduire facilement (essayez pour voir...) :

$$\begin{aligned} n \ \equiv \ 0\ [3] \ \Leftrightarrow \ \sum_{i=0}^{k} n_i \ &\equiv \ 0\ [3]\,, \\ n \ \equiv \ 0\ [11] \ \Leftrightarrow \ \sum_{i=0}^{k} (-1)^i n_i \ &\equiv \ 0[11]. \end{aligned}$$

Ecrire les fonctions `Multde3` et `Multde11` qui testent si un entier `n` est multiple de 3 et de 11.

7. **Partition d'un entier**
On appelle partition d'un entier n un ensemble d'entiers strictement positifs $\{p_0, p_1, \ldots, p_k\}$ tel que $\sum_{i=0}^{k} p_i = n$. Par exemple :

$$\begin{aligned} 3 &= 3 \\ &= 3+2 \\ &= 1+1+1 \end{aligned}$$

Le but de cet exercice est d'afficher toutes les partitions d'un entier n positif. Pour éviter de considérer les permutations d'un même ensemble $(1+2 = 2+1 = 3)$, on impose un ordre sur l'affichage. En choisissant l'ordre croissant, $1+2$ sera affiché mais pas $2+1$. Si un entier i apparaît dans une somme, aucun entier inférieur n'apparaît à sa droite.

Ecrire une fonction `Part(int n,int min,int tab[],int long)` affichant les partitions de n ne contenant pas d'entiers inférieurs à `min`. La partition en cours d'élaboration sera stockée dans le tableau `tab`. La variable `long` indique le nombre d'éléments déjà déterminés dans le tableau `P`.