الكلية متعددة التخصصات تارودانت
ⵜⴰⴽⵍⵉⵜ ⵜⴰⵙⵜⵉⵎⵣⵉⵏⵜ ⵏ ⵜⵓⵙⵏⴰⵡⵜ
FACULTE POLYDISCIPLINAIRE TAROUDANT

TP2 : Les fonctions récursives

Enseignants	Pr. TAHA
Tags	Récursivité TP Tri et recherche
Année universitaire	2024-2025
Section	MIP_GI_S2

Objectifs :

1. Implémenter et analyser des fonctions récursives pour des problèmes numériques et de structures de données.
2. Comparer approches récursive et itérative.
3. Éviter les pièges de la récursion (profondeur d'appel, cas de base).

Exercice 1 : Exponentiation rapide

> **But :** calculer x^n en $O(\log n)$ plutôt qu'en $O(n)$.

Écrire une fonction récursive d'entête `def puissance(x,n)` qui retourne la valeur de x^n en utilisant la formule suivante:

$$puissance(x,n) = \begin{cases} x & \text{si } n = 1 \\ puissance(x^2, n/2) & \text{si } n \text{ est pair} \\ x \times puissance(x^2, (n-1)/2) & \text{si } n > 2 \text{ est impair} \end{cases}$$

Exercice 2 : Conversion binaire ↔ décimal

- Ecrire une fonction récursive d'entête `def base2(n: int) -> list[int]:` qui permet de convertir un entier naturel n dans le système binaire. La suite binaire est stockée dans une liste.

```
def base2(n: int) -> list[int]:
    """
    Retourne la représentation binaire de n sous forme de liste de bits.
    """
```

- Ecrire une fonction récursive d'entête `def base10(a):` qui retourne la valeur décimale d'une suite binaire repassée comme argument.

```
def base10(a):
    """
    Si a est un entier (ex. 1011), le traiter comme chaîne de caractères.
    Si a est une liste [1,0,1,1], la convertir en entier.
    """
```

Tests :

- `base2(6) → [1,1,0]`
- `base10([1,1,0]) → 6`
- `base10(1111) → 15`

Exercice 3 : Opérations récursives sur les listes

> On interdit l'usage de `append`, `len`, `sort`, `reverse`, etc.

1. `def taille_liste(L) → int` : retourne le nombre d'éléments.
2. `def sum_liste(L) → float` : calcul la somme de tous les éléments de la liste.
3. `def aff_liste_inv(L) → None` : affiche les éléments dans l'ordre inverse.

Exercice 4 : Opérations récursives sur les chaînes

Ecrire les fonctions récursives suivantes :

- `def strcpy(src: str, dest: str) → str` : copie récursive caractère par caractère.
- `def strcpm(ch1: str, ch2: str) → int` : compare deux chaînes ; retourne
  - `1` si `ch1 > ch2`
  - `0` si `ch1 == ch2`
  - `1` si `ch1 < ch2`
- `def anagramme(ch1: str, ch2: str) → bool` : détermine si `ch1` et `ch2` sont anagrammes.

> Une **anagramme** est un mot ou une expression obtenu(e) en **réarrangeant** toutes les lettres d'un autre mot ou d'une autre expression, sans en ajouter ni en retirer. L'idée est que, lettre à lettre, les deux chaînes partagent exactement le même ensemble de caractères.
>
> **Exemple simple :**
>
> `anagramme("algorithme", "logarithme")` retourne `True`
>
> `anagramme("chien", "nich")` retourne `false`

Exercice 5 : Refaire tous les exercices du TD

- Refaire tous les exercices du TD

- Refaire tous les exercices du TD
- Refaire tous les exercices du TD
- Refaire tous les exercices du TD