1 Recherche de motif, de sous-chaine

Exercice 1. Niveau 2. Ecrire une fonction `def rechMot(chaine, mot)` qui cherche la présence de `mot` à l'intérieur de `chaine`, et qui renvoie le premier indice à partir duquel `mot` est présent dans `chaine` en cas de présence, et qui renvoie `None` sinon.

Un algorithme naïf est attendu, avec deux boucles imbriquées.

Il est tout à fait permis, si on préfère, d'écrire une première fonction `def presentAPartirDe(chaine, mot, i)` et qui renvoie `True` si `mot` est présent dans `chaine` à partir de l'indice `i`, et `False` sinon.

On s'interdira en revanche (même si cela marcherait très bien !) de se contenter d'utiliser des slices (tranches). Autrement dit, la fonction suivante ne constitue pas une solution acceptable :

```
def presentAPartirDe(chaine, mot, i):
    return chaine[i:i+len(mot)] == mot
```

2 Palindrome

Exercice 2. Niveau 1. Un palindrome est une chaîne de caractères qui lue de droite à gauche est identique à sa lecture de gauche à droite. Ecrire donc une fonction d'entête `def estPalindrome(chaine)` qui teste si une chaine donnée est ou n'est pas un palindrome. (Ici encore, on ne se permettra que la lecture caractère par caractère de `chaine`, ainsi du code tel que `return chaine[::-1] == chaine` n'est pas une réponse acceptable…)

3 Suite de Conway

Voici les premiers termes de la fameuse suite de Conway : 1, 11, 21, 1211, 111221

Question. Quel en est le terme suivant ? (Indication : lire pour chaque terme les chiffres à voix haute)

Exercice 3. Niveau 3. Ecrire une fonction d'entête `def suivant(terme)` qui prend en argument un terme de la suite de Conway (sous la forme d'un entier) et qui renvoie le terme suivant de la suite de Conway. (Sous la forme d'un entier encore)

Exercice 4. Niveau 3. On reprend l'idée de la suite de Conway, mais on va raisonner selon une autre base que la base 10. Par exemple, en base 2, la suite de Conway (les nombres qui suivent sont écrits en base 10, attention !) toujours en partant de 1 donnerait, pour ses premiers termes : 1, 3, 5, 59 (en effet, la suite de Conway en base 2 donnerait, pour ses premiers termes : 1, 11, 101, 111011 qui sont les écritures en base 2 des entiers respectifs 1, 3, 5, 59)

Reprendre les fonctions `convertit` de `deBaseB` de la feuille précédente, et écrire une fonction d'entête `def suivant2(terme, b)` et qui prend en argument un terme de la suite de Conway, l'écrit en base `b`, obtient l'écriture du terme suivant en base `b` et renvoie la valeur de celui-ci, sous forme d'un entier. Exemples :

```
>>> suivant2(5, 2)
    59
>>> suivant2(7, 3)
    49
>>> suivant(49, 3)
    376
```