Feuille_info 3 : Exercices rapides.

Sujet 1 :

1) Ecrire une fonction `nombre1` prenant en argument une liste `L` ne contenant que des 0 et des 1 et renvoyant le nombre de 1 dans `L`.
2) Ecrire une fonction `serie1` prenant en argument une liste `L` ne contenant que des 0 et des 1 et renvoyant la longueur de la première série du premier élément.
   Exemples : `serie1([1, 1, 1, 0, 0, 1, 1])` renvoie `3`
   `serie1([0, 0, 1, 1, 0, 0, 1, 1])` renvoie `2`
3) Ecrire une fonction `serie2` prenant en argument une liste `L` ne contenant que des 0 et des 1 et renvoyant la longueur de la deuxième série de la liste.
   Exemples : `serie2([1, 1, 1, 0, 0, 1, 1])` renvoie `2`
   `serie2([0, 0, 1, 1, 1, 0, 0, 1, 1])` renvoie `3`

Sujet 2 :

1) Ecrire une fonction `Dist` prenant en argument une liste `L` et renvoyant une liste contenant tous les élements distincts de `L`.
   Exemple : `Dist([4, 1, 2, 3, 3, 4, 1])` renvoie `[4, 1, 2, 3]`
2) Ecrire une fonction `Nb_elt` prenant en argument une liste `L` et renvoyant la liste précédente ainsi qu'une liste contenant le nombre de fois où un élément est présent dans `L`.
   Exemple : `Nb_elt([4, 1, 2, 3, 3, 4, 1])` renvoie `[4, 1, 2, 3], [2, 2, 1, 2]`
3) Ecrire une fonction prenant en argument deux listes et qui renvoie `True` si elles sont permutations l'une de l'autre, et `False` sinon.

Sujet 3 :

1) Ecrire une fonction prenant en argument L (éventuellement `a` et `b`) qui renvoie `True` si toutes les valeurs de L sont entre a et b, `False` sinon (a et b des réels)
2) écrire une fonction prenant en argument L et qui renvoie la liste du nombre d'occurence de chaque valeur entre a et b (a et b des entiers, $a < b$) (on suppose que toutes les valeurs de L sont dans cet intervalle), la première valeur de la liste retournée est le nombre d'occurrence de a, et la dernière valeur le nombre d'occurrences de b
3) Si la fonction précédente n'a pas été écrite avec une seule boucle, la refaire avec une seule boucle
4) Ecrire à l'aide de la fonction précédente, une fonction qui renvoie la liste L triée.

Sujet 4 :

1) Ecrire une fonction `palindrome(mot)` qui prend pour paramètre `mot` une chaine de caractères et qui renvoie `True` si `mot` est un palindrome et `False` sinon.
2) Ecrire une fonction `anagramme(mot1, mot2)` qui prend pour paramètres `mot1` et `mot2` deux chaines de caractères et qui renvoie `True` si `mot1` est une anagramme de `mot2` et `False` sinon.
3) Ecrire une fonction `liste_anagrammes(mot)` qui prend pour paramètre `mot` une chaine de caractères et qui renvoie la liste de toutes les anagrammes de `mot`.

Précision :

Un **palindrome** est un mot qui reste le même si on le lit de gauche à droite ou de droite à gauche.

Exemples de palindromes : radar, kayak, ici, ressasser, AABBAA.

Une **anagramme** est un mot obtenu en permutant les lettres d'un mot.

Ici les mots sont justes des chaines de caractères, ils n'ont pas à avoir de sens.

Par exemple, la liste des 12 anagrammes de ANNE est :

AENN , ANEN, ANNE, EANN, ENAN, ENNA, NAEN, NANE, NEAN, NENA, NNAE, NNEA

Sujet 5 : Pour n un entier naturel non nul et k un entier vérifiant $1 \leqslant k < n$,

`n%k` renvoie le reste de la division euclidienne de n par k.

On dit que k est un diviseur strict de n lorsque `n%k` est nul.

On note S_n la somme des diviseurs stricts de n, par exemple $S_3 = 1$, $S_4 = 1+2 = 3$, $S_{12} = 1+2+3+4+6 = 16$.

1) Ecrire une fonction `S` prenant en argument un entier n et qui renvoie la valeur de S_n

On appelle nombre parfait un nombre entier n vérifiant $S_n = n$. Exemple : 6 est parfait car $S_6 = 1 + 2 + 3$.

2) Ecrire une fonction `Nb_parf` prenant en argument un entier N et qui renvoie le nombre de nombres parfaits dans un intervalle $[\![1; N]\!]$.
3) Ecrire une fonction renvoyant le $k^{\text{ième}}$ nombre parfait *(la fonction ne prend que k en argument).*