TD nº 1 - Quelques algorithmes simples

Exercice 1. *Écritures binaires*

1. Écrire une fonction `zero_un(s)` qui teste si une chaîne de caractères n'est formé que des caractères `0` et `1`.

2. Écrire une fonction `eval(s)` qui renvoie l'entier représenté par une chaîne de caractères en binaire (par exemple `eval("1011")` renvoie `11`).

3. Écrire un programme qui demande à l'utilisateur d'entrer une chaîne de caractères puis utilise les fonctions précédentes pour afficher la valeur de l'entier représenté par cette chaîne si elle ne contient que des `0` et des `1`, ou écrit « erreur » sinon.

Exercice 2. *Codage et décodage*

On rappelle que les caractères sont codés par des entiers entre 0 et 255, et que les caractères de 'A' à 'Z' sont codés par des entiers successifs.
On pourra utiliser dans cet exercice les fonctions `ord(c)` qui renvoie l'entier codant un caractère `c`, et `chr(i)` qui renvoie le caractère codé par l'entier `i`.

1. Écrire une fonction qui prend en argument un caractère (une chaîne de longueur 1) et renvoie ce caractère si c'est une lettre majuscule. Sinon la fonction renvoie le caractère correspondant à l'entier 0.

2. Écrire une fonction `compte(s,c)` qui compte le nombre d'occurrences d'un caractère `c` dans une chaîne `s`.

3. En utilisant la fonction `compte`, écrire une fonction `nb_lettres(s)` qui compte le nombre d'occurrences de chaque lettre majuscule dans la chaîne `s` et renvoie le résultat sous la forme d'un tableau de 26 cases.

4. Combien de fois la chaîne `s` est-elle parcourue lorsque l'on exécute la fonction `nb_lettres` ?

5. Ré-écrire la fonction `nb_lettres` pour qu'elle ne parcoure la chaîne `s` qu'une seule fois.

Exercice 3. *Elu par cette crapule, Esope reste et se repose*

Un *palindrome* est un mot qui se lit pareil de gauche à droite ou de droite à gauche (par exemple « bob » et « laval » sont des palindromes).

1. Écrire une fonction `palindrome(s)` qui teste si la chaîne de caractères `s` est un palindrome.

Exercice 4. *Tri rapide*

On va maintenant trier des tableaux d'entiers...

1. Écrire une fonction `verifie(tab)` qui vérifie si un tableau est déjà trié.

Le principe de l'algorithme *quick sort* consiste à choisir une valeur v (appelée *pivot*) dans le tableau `tab` puis à modifier le tableau de telle sorte que la valeur v se trouve à un indice j, que toutes les valeurs avant l'indice j soient inférieures à v, et que toutes les valeurs après l'indice j soient supérieures à v.
On recommence alors sur les portions du tableau entre les indices 0 et $(j-1)$ et entre les indices $(j+1)$ et `len(tab)`, et ainsi de suite jusqu'à ce que le tableau soit trié.

2. Exécuter à la main le fonctionnement de l'algorithme sur le tableau `[3, 0,11, 2, 9, 2, 8]` en prenant à chaque fois la première case du tableau considéré comme pivot.

3. Écrire une fonction `pivot(tab, i)` qui modifie le tableau `tab` de telle sorte que toutes les valeurs inférieures à `tab[i]` soient placées à gauche de `tab[i]` et que toutes les valeurs plus grandes soient placées à droite.

4. Écrire une fonction récursive `quick_sort(tab, i, j)` qui trie la partie du tableau `tab` entre les indices `i` et `j` à l'aide de l'algorithme *quick sort*.