Algorithmes de tri

Les données structurées sont organisées au sein de bases de données.
Rechercher efficacement une information dans un tableau trié d'un million de valeurs va 50 000 fois plus vite que dans ce même tableau non trié ! Il est donc nécessaire de classer les données selon certains critères.
On réalise alors des opérations de tri qui doivent être les plus rapides possibles.

Exercice 1

Découpez les cellules du tableau qui se trouve en bas du polycopié. Tournez les faces cachées et mélangez-les. Le but est de les remettre dans l'ordre en suivant une méthode de votre choix.

Règles

- Vous ne pouvez retourner que deux cartes simultanément.
- Vous pouvez inverser ou non les positions des deux cartes avant de les replacer face retournée.
- Vous ne devez pas mémoriser les valeurs inscrites sur les cartes au fur et à mesure de votre travail.
- Vous comptera le nombre de paires de cartes que vous aurez retournées et le nombre de paires de cartes que vous avez inversées.

Exercice 2 : Consulter les articles suivants : Tri stupide Tri par insertion Tri à bulles Tri rapide

Exercice 3 Tri rapide en Python

On utilise des listes. Attention les valeurs sont numérotées à partir de 0.

>>> a=[1,2]
>>> b=[3,4]
>>> a+b
[1, 2, 3, 4]
>>> b[1]
4
>>> b[0]
3
>>> a.append(7)
>>> a
[1, 2, 7]
	On a nommé a la liste [1,2]
	On a nommé b la liste [3,4]
	La commande + est utilisée pour concaténer deux listes
	La deuxième valeur de la liste b est 4
	
	La première valeur de la liste b est 3
	
	La commande append ajoute une valeur à la fin d'une liste

La fonction quicksort est **récursive**, c'est-à-dire qu'elle fait appel à elle-même (mais sur des listes plus petites)

```
def quicksort(t):
    if t == []:
        return []
    else:
        pivot = t[0]
        t1 = []
        t2 = []
        for x in t[1:]:
            if x<pivot:
                t1.append(x)
            else:
                t2.append(x)
    return quicksort(t1)+[pivot]+quicksort(t2)
```

On définit une fonction quicksort
si la liste est vide elle ne change pas

sinon
on prend comme pivot le premier élément de la liste
t1 et t2 sont au départ deux listes vides

on parcourt tous les éléments de t à partir de la position 2, si cet élément est inférieur au pivot, on le place à la fin de t1
sinon on le place à la fin de t2
on renvoie la liste formé de la concaténation du tri de t1, du pivot et du tri de t2

>>> import random
>>> random.randrange(0, 10)
9
>>> l = [random.randrange(0, 10) for i in range(20)]
>>> l
[8, 8, 6, 3, 9, 1, 5, 5, 3, 8, 7, 1, 7, 1, 7, 5, 5, 7, 2, 4]
	Importer la bibliothèque de commandes aléatoires
	générer un nombre entier aléatoire entre 0 et 10
	
	on crée une liste de 21 nombres entiers aléatoires

Utiliser le programme quicksort pour trier la liste aléatoire.

1	2	3	4	5	6	7	8	9	10