T.NSI **TP du chapitre 4 : Mise au point de programmes**

L'ensemble du travail est à effectuer dans un seul fichier `Chap4TP.py`

Question 1
Définir une fonction `occurrences(t:list) → dict`
qui renvoie un dictionnaire des occurrences d'un tableau `t`.
Chaque valeur v contenue dans t est une clé de ce dictionnaire.
Et ce dictionnaire associe à la valeur v le nombre d'occurrences de cette valeur dans t.

Exemple : `occurrences([10,20,30,20,10,20])` doit renvoyer `{10: 2, 20: 3, 30: 1}`

Question 2
1) Définie une fonction `compare_dico(d1:dict, d2:dict)→bool`
qui renvoie True si et seulement si les dictionnaires d1 et d2 contiennent exactement les mêmes clés et les mêmes valeurs.

Exemple : `compare_dico({10:2, 20:3}, {20:3,10:2})` renvoie `True`
mais `compare_dico({10:2}, {20:3,10:2})` renvoie `False`

2) Simplifier cette fonction `compare_dico` en utilisant des **assertions** pour détecter une éventuelle différence entre d1 et d2

Question 3
Définir une fonction `teste_tri(t)` qui teste la fonction `tri(t)` sur un tableau `t`
Cette fonction utilisera des assertions : si le test est incorrect, l'exécution doit déclencher une AssertionError.

On pourra utiliser le tri par sélection

```
def tri(t):
    for i in range(len(t)-1):
        m = i
        for j in range(i + 1, len(t)):
            if t[j] < t[m]:
                m = j
        t[i],t[m] = t[m],t[i]
```

ou le tri de python

```
def tri(t):
    t.sort()
```

Question 4
1) Définir une fonction `random_tab(n,a,b)` qui renvoie un tableau de taille `n` d'entiers aléatoires compris entre `a` et `b`.

2) Compléter le code suivant pour effectuer le test de tris sur des tableaux aléatoires « bien choisis ».

```
for n in range(...):
    teste_tri(random_tab(n, . . . , . . . ) )
    teste_tri(random_tab(n, . . . , . . . ) )
    teste_tri(random_tab(n, . . . , . . . ) )
```