TP n° 1 : Diviser pour régner

Rappels :

Récursivité :

- cas d'arrêts, paramètre...
- à chaque appel récursif, on se rapproche d'un cas d'arrêt
- si cas d'arrêt solution
- sinon appel(s) récursif(s)

Les listes en python :

- Construction : append et extend sont 2 procédures qui modifient une liste :
  Ajout d'une valeur en fin de liste : `L.append(L[0])`
  Ajout de tous les éléments de L1 en fin de L : `L.extend(L1)`
- Affectation :

```
L1=L2    # L1 pointe vers L2 , même zone mémoire (shallow copy)
L1=L2[:] #L1 et L2 occupent des espaces mémoire distincts (deep copy)
```

- Test d'égalité : simplement L1==L2
- Slices : L[i:j] 'tranche de L' de l'indice i à l'indice j-1, on a :
  `L[:2] ==[L[0],L[1]]`
- Tri de liste:

```
L1=sorted(L)  #L1 est une nouvelle liste,L est conservée
L.sort()      #L est modifiée en place
```

Exercice 1 : Recherche d'un élément dans un tableau

- Méthode : algorithme récursif de paramètre la taille du tableau
- Écrire une fonction qui retourne un booléen vrai si et seulement si la valeur elt est présente dans le tableau d'entiers tab. On écrira 2 versions :
  - Cas général : tableau quelconque d'entiers
  - Cas particulier : adapter l'algorithme au cas d'un tableau rangé dans l'ordre croissant.

Tests : faciles en python grâce à la fonction in. Effectuer de nombreux appels de la fonction à tester avec une valeur aléatoire à rechercher dans des listes aléatoires :

```
for i range(100) :
    nb=randrange(1,10)
    #génération d'une liste aléatoire de nb nombres
    L=[randrange(0,9) for p in range(0,nb)]
    L.sort()  # seulement pour le cas particulier du tableau rangé
    elt=randrange(0,9)
     trouve1= elt in L
    trouve2= appel fonction
    if trouve1!=trouve2 :
        print('Pb')
```

.

Exercice 2 : Tri fusion

- Méthode : la procédure de tri fusion (TF) se décompose en 3 parties : (algorithme récursif vu en cours):

  DIVISER : découper le tableau en 2 parties égales.
  REGNER : trier récursivement les 2 sous tableaux (2 appels récursifs de `tri_fusion`)
  COMBINER : combiner les 2 tableaux obtenus (procédure `fusionner`)

- Dans la suite, on va écrire successivement diverses versions de `fusionner` et de `tri_fusion` et les tester .
- Exemples :

  `fusionner([0,1,3,5,5,6],[1,1,3,4])=[0,1,1,1,3,3,4,5,5,6]`

  `fusionner([0,1],[2,6])=[0,1,2,6]`

  `fusionner([5,6,6],[1])=[1,5,6,6]`

- Tests de `fusionner` : faciles en python grâce à la fonction `sorted()` qui permet de trier une liste . Tester à la main sur quelques exemples.

- Tests de `tri_fusion` : faciles en python grâce à la fonction `sorted()` . La fonction suivante prend en paramètre le nom de la fonction de tri à tester (`tri_a_tester`) et produit un affichage en cas de problème :

```
def test_tri( tri_a_tester ):
   for i range(100) :
      nb=randrange(1,10)
      #génération d'une liste aléatoire de nb nombres
      L=[randrange(0,9) for p in range(0,nb)]
      L1=sorted(L) #liste triée avec la fonction built in
      L2=tri_a_tester(L) #liste triée avec la fonction à tester
      if L1!=L2 :
           print('Pb :')
           print(L1)
           print(L2)
 #appel de la fonction  test_tri
test_tri(tri_fusion0(L))
```

Attention : Pour fonctionner la fonction `tri_a_tester` doit prendre pour seul argument la liste à trier !

1. Version itérative :

   Écrire la fonction itérative `fusion0(L1,L2)` qui, prend en arguments deux listes déjà triées L1 et L2 renvoie la liste résultat de la fusion de L1 et de L2 et la tester.

.

Écrire alors la fonction `tri_fusion0` et la tester.

2. Version récursive :

   Diverses possibilités :

   - Première version : implémenter l'algorithme récursif suivant pour écrire la fonction récursive `fusion1`

     si (l'un des tableaux à fusionner T1 ou T2 est vide) le résultat est l'autre tableau

     sinon si T1[0]<T2[0] le résultat commence par T1[0] et se poursuit avec la fusion de T1[1:] et de T2

     sinon le résultat commence par T2[0] et se poursuit avec la fusion de T1 et de T2[1:]

   Écrire alors la fonction `tri_fusion1` et la tester.

   - Deuxième version : avec fonction auxiliaire :

   Écrire une fonction récursive `inserer(x,L)` qui prend en argument un entier x et une liste L triée par ordre croissant et insère x dans la liste de manière à conserver l'ordre.

   Écrire une fonction récursive `fusion2(L1,L2)` (qui utilise inserer...) .

   Écrire alors la fonction `tri_fusion2` et la tester.

.