# 4TPM205U: Algorithmique des tableaux: feuille 4

## Tableaux : 2

## Travaux dirigés

**Pour chacun des exercices, indiquer, en fonction du nombre initial d'éléments du tableau, le nombre total d'éléments décalés.**

**Exercice 1** – Écrire une fonction `inserer(t, n, elt, k)` qui, étant donné un tableau `t` contenant `n` éléments (avec `n` < `len(t)`) insère un élément `elt` à la position `k` avec `k` $\geq 0$.
Si `k` $\geq$ `n`, l'élément sera ajouté à l'indice `n`. L'ordre initial des éléments du tableau sera conservé.
La fonction doit renvoyer le nouveau nombre d'éléments du tableau.

*Exemple* :
si `len(t)` vaut 10 et `t = [5,4,3,12,0,4,7,2,None,None]`, l'appel `inserer(t,8,6,4)` doit

- insérer l'élément 6 à l'indice 4, après l'insertion le tableau contiendra `[5,4,3,12,6,0,4,7,2,None]`,
- renvoyer le nouveau nombre d'éléments dans `t`, c'est à dire 9.

**Exercice 2** – Écrire une fonction `supprimer(t, n, k)` qui supprime l'élément situé à la position `k` du tableau `t` contenant `n` éléments en supposant $0 \leq$ `k` < `n`. L'ordre initial des éléments du tableau sera conservé. La fonction doit renvoyer le nouveau nombre d'éléments dans le tableau.

**Exercice 3** – Écrire une fonction `supprimerPremiereOccurrence(t, n, elt)` permettant d'enlever du tableau `t` la première occurrence d'un élément `elt` passé en paramètre (le tableau `t` ne sera pas modifié si `elt` n'appartient pas à `t`). L'ordre initial des éléments du tableau sera conservé. La fonction doit renvoyer le nouveau nombre d'éléments dans le tableau.

**Exercice 4** – Soit `t` un tableau contenant `n` nombres rangés dans l'ordre croissant, avec `n` < `len(t)`.
Écrire une fonction `insererOrdonne(t, n, elt)` qui insère un nouvel élément `elt` dans `t` en respectant l'ordre croissant et qui renvoie le nouveau nombre d'éléments dans le tableau.

*Exemple* :
si `len(t)` vaut 10 et `t =[2,4,7,10,15,20,25,None,None,None]`, l'appel `insererOrdonne(t,7,5)` va renvoyer 8 (nombre d'éléments dans `t` après l'insertion) et va aussi modifier le contenu de `t` en `[2,4,5,7,10,15,20,25,None,None]`.

**Exercice complémentaire 1** – Écrire une fonction `estSection(t, nt, s, ns)` qui prend en paramètre deux tableaux d'entiers `t` et `s` contenant respectivement `nt` et `ns` éléments et qui renvoie `True` si `s` correspond à une *section* du tableau `t` et `False` sinon.

Une *section* est une suite éventuellement vide d'éléments contigus dans un tableau. Par exemple, cette fonction renvoie `True` pour `t = [5, 1, 2, 3, 1, 2, 1]` et `s = [1, 2]`. Elle renvoie `False` pour `t = [1, 2, 3, 4, 1]` et `s = [1, 3]`.

**Exercice à traiter en travail personnel pour la séance de TD suivante.**

**Exercice 5** – Soit la fonction `mystere` suivante :

```
1 def mystere(t,n,x):
2     continuer = True
3     while n>0 and continuer:
4         nbelem = supprimerPremiereOccurrence(t,n,x)
5         if nbelem == n :
6             continuer = False
7         else:
8             n = nbelem
9     return n
```

1. Soit `t1=[2,-7,4,5,12,10,4,2,4,-18]` un tableau contenant `10` éléments. Simulez l'exécution de `mystere(t1,10,4)` en complétant le tableau suivant pour montrer l'évolution des variables `n`, `continuer`, et `nbelem`. Précisez également le contenu de `t1` à chaque étape.
   `t1=[2,-7,4,5,12,10,4,2,4,-18]`

| `n` | | |
|---|---|---|
| `continuer` | | |
| `nbelem` | | |

2. Quel sera en général le résultat d'un appel de `mystere` ?
3. Quel est le nombre d'éléments du tableau décalés dans le meilleur et dans le pire des cas ? Quel est le nombre de comparaisons concernant des éléments du tableau dans le meilleur et dans le pire des cas ?