FEUILLE N° 2 : INVARIANTS ET TESTS

Exercice 1 : Partition d'un tableau

Étant donné une liste de réels T, on considère le slice[1] T[deb :fin+1] . La première valeur du slice est choisie comme pivot :

pivot= T[deb]

Partitionner le tableau consiste à placer le pivot à sa place définitive dans le tableau, en permutant tous les éléments de telle sorte que tous ceux qui sont inférieurs au pivot soient à sa gauche et que tous ceux qui sont supérieurs au pivot soient à sa droite.

Précisément, on a placé en début de t les valeurs strictement inférieures T[deb] puis le pivot T[deb] puis les valeurs supérieures ou égales au pivot.
La spécification en triplet de Hoare est :

```
#PE: 0<=deb<=fin<len(T)
#partition(t,DEB,FIN,a)
#PS: DEB ≤ a ≤ FIN et  ∀J  (0 ≤ J <a  → T[J]< pivot) et t[a]=pivot et ∀J (a< J ≤ FIN → pivot ≤ T[J])
```

Ecrire une fonction qui renvoie l'indice de la PREMIERE occurrence du pivot dans t.

Méthode 1 : On utilise le principe du drapeau hollandais en rangeant les nombres en 3 catégories : inférieurs stricts au pivot, égaux au pivot , strictement supérieurs au pivot.

Méthode 2 :

- ✔ On range d'abord le slice t[deb+1,fin+1] en plaçant à gauche les valeurs **strictement** inférieures au pivot puis les valeurs supérieures **ou égales** au pivot. On utilise 2 variables :
  - d : indice de la première case après la zone connue de valeurs <= au pivot,
  - f : indice de la première case avant la zone connue de valeurs > pivot.

  A chaque étape, on réduit la zone inconnue comprise entre les bornes d et f par comparaison avec le pivot.
- ✔ Pour terminer, on place le pivot à sa place définitive (échange entre t[deb] et une autre case)

Pour les 2 méthodes :

1. Déterminer l'invariant de boucle
2. Écrire la fonction **`partition(t,deb,fin)`**
3. Tester avec une fonction **`test_partition`** (sur le modèle de la fonction test_tri).

1 Le slice t[a,b] est la sous-liste de t pour les indices allant de a à b-1

Exercice 2 : Fusion de deux tableaux

On dispose d'une liste L telle que les 2 slices `L[ a:b+1]` et `L[b+1:c+1]` contiennent chacun des valeurs croissantes . On veut modifier L de telle sorte qu'en sortie le slice `L[a:c+1]` contienne rangées dans l'ordre croissant toutes les occurrences des valeurs initialement présentes dans `L[a,c+1]` . On dit qu'on a fusionné les 2 slices `L[ a:b+1]` et `L[b+1:c+1]`.

- Exemples :

```
fusion([0,1,3,5,5,6,1,1,3,4],0,5,9)=[0,1,1,1,3,3,4,5,5,6]
fusion([0,1,2,6],0,1,2)=[0,1,2,6]
fusion([5,6,6,1],0,2,3)=[1,5,6,6]
fusion([1],0,0,0)=[1]
```

1. Spécifier en triplet de Hoare.
2. On utilise un tableau auxiliaire qu'on remplit en parcourant les deux slices séquentiellement. On pioche le nombre le plus petit et on le stocke dans le tableau auxiliaire... Déterminer l'invariant de boucle.
3. Écrire la fonction **`fusion(L,a,b,c)`**
4. Justifier la bonne terminaison de l'algorithme.
5. Tester en créant une fonction `test_fusion` (sur le modèle de la fonction **`test_tri`**).