 # TD 3 - Premières étapes de test

On considère une classe Tableau permettant de stocker des entiers dans un tableau trié. Le nombre d'éléments du tableau est stocké dans un attribut taille et la longueur du tableau dans un attribut capacité. La taille est toujours supérieure ou égale à 0, la capacité toujours strictement positive, et la taille inférieure ou égale à la capacité. En particulier, on ne peut pas appeler le constructeur de la classe avec un argument négatif.

```
public class Tableau {
 private int tab[];
 private int taille;
 private int capacite;

 public boolean supprimer(int val) {
  int i = 0;
  while(i < taille && tab[i] != val) {
   i++;
  }
  if(i == taille) {
   return false;
  }
  for(int j = i; j < taille-1; j++) {
   tab[j] = tab[j+1];
  }
  tab[taille-1] = 0;
  taille = taille-1;
  return true;
 }
```

```
 public boolean inserer(int val) {
  int i;
  if(taille == capacite) {
   return false;
  }
  for(i = taille; i > 0 && tab[i-1] > val; i--) {
   tab[i] = tab[i-1];
  }
  tab[i] = val;
  taille = taille+1;
  return true;
 }
}
```

Nous cherchons à tester les deux méthodes insert et delete de cette classe, en interne, c'est à dire en ayant accès à tous les attributs. Si la table n'est pas pleine, la méthode insert ajoute l'élément passé en argument à la table en préservant l'ordre et renvoie true. Si la table est pleine, elle renvoie false. La méthode delete supprime une occurrence de l'élément passé en argument et renvoie true. Elle renvoie false si l'élément n'était pas présent.

1. Proposer un ensemble de tests pour la fonction delete.

2. Pour les modifications suivantes du programme, indiquer si la modification introduit une erreur et, lorsque cela est possible, donner un cas de test pour révéler l'erreur.

  * (a) Nous modifions la condition if par i == size-1.

  * (b) Nous modifions la condition d'arrêt de la boucle for par j < size.

  * (c) Nous modifions l'initialisation de la boucle for par j=i+1.

  * (d) Nous changeons size par capacity dans les conditions while et if.

  * (e) Nous supprimons la ligne tab[size-1] = 0.

* (f) Nous effectuons les modifications (d) et (e) ensemble.

3. Pour chacun des tests suivants, donner une implémentation de la fonction insert qui échoue à ce test, en apportant de simples modifications au code donné.

```

tab size capacity val

Test 1 [3,5,6,8] 4 4 7

Test 2 [2,4,5] 3 8 1

```

4. Nous testons maintenant ces méthodes depuis l'extérieur de la classe. De quels observateurs aurions-nous idéalement besoin pour pouvoir détecter autant de défauts qu'avec les tests internes ?

5. Nous ajoutons les méthodes suivantes à la classe Tableau, Quels aspects de l'implémentation ne peuvent plus être observés ? Peuvent-ils être testés ? Comment ? :

```

int size() int capacity()

boolean present(int val)

```

6. Pour deux des tests proposés dans la question 1, construisez la séquence d'appels de méthodes (le scénario de test) pour exécuter ce test, depuis l'initialisation des objets jusqu'à la vérification du résultat obtenu.

1)

Pour tester la fonction supprimer, nous devons nous assurer de couvrir toutes les possibilités possibles

Voici un ensemble de tests :

1. **Suppression dans un tableau vide** :
    - Tableau : []
    - taille : 0

- Valeur à supprimer : 5
- Résultat attendu : false

2. **Suppression d'un élément qui n'est pas présent** :
    - Tableau : [1, 2]
    - taille : 2
    - Valeur à supprimer : 3
    - Résultat attendu : false
3. **Suppression du premier élément** :
    - Tableau : [1, 2]
    - taille : 2
    - Valeur à supprimer : 1
    - Résultat attendu : true
4. **Suppression du dernier élément** :
    - Tableau : [1, 2]
    - taille : 2
    - Valeur à supprimer : 2
    - Résultat attendu : true
5. **Suppression d'un élément au milieu** :
    - Tableau : [1, 2, 3, 4, 5]
    - taille : 5
    - Valeur à supprimer : 3
    - Résultat attendu : true
6. **Suppression de tous les éléments d'un tableau** :
    - Suppression répétée jusqu'à obtenir un tableau vide. Par exemple :
        - Tableau initial : [1]
        - Valeur à supprimer : 1
        - Résultat attendu : [0]

2)

(a)

Des erreurs apparaissent avec cette modification. A cause de cela, le premier et le dernier élément ne pourront pas être supprimer correctement.

Cas de test :

- Tableau : [1, 2, 3]
- taille : 3
- Valeur à supprimer : 3
- Résultat attendu : true
- Résultat obtenu : false

(b)

Si on change la condition d'arrêt à j < taille, la boucle accèderait à un index hors des limites du tableau, causant une erreur d'exécution.

Cas de test :

- Tableau : [1, 2, 3, 4]
- taille : 4
- Valeur à supprimer : 2
- Résultat attendu : [1, 3, 4, 0], taille : 3.

Avec la modification, il y aurait une tentative d'accès à l'indice taille, causant un dépassement de tableau.

(c)

Si on initialise j = i + 1, l'élément juste après l'élément supprimé sera supprimé a la place

Cas de test :

- Tableau : [1, 2, 3, 4]
- taille : 4
- Valeur à supprimer : 2
- Résultat attendu : [1, 3, 4, 0], taille : 3.
- Résultat obtenu : [1, 2, 4, 0].

(d)

Cela introduit une erreur. capacite indique la capacité maximale du tableau et non sa taille réelle.

Cas de test :

- Tableau : [1, 2, 3]
- taille : 3
- capacite : 5
- Valeur à supprimer : 4
- Résultat attendu : false.

Avec la modification, la boucle pourrait parcourir les indices vides du tableau, menant à des résultats imprévisibles.

(e)

En la supprimant, le tableau pourrait contenir des valeurs anciennes après la suppression.

Cas de test :

- Tableau : [1, 2, 3, 4]
- taille : 4
- Valeur à supprimer : 4
- Résultat attendu : [1, 2, 3, 0].
- Résultat obtenu : [1, 2, 3, 4], bien que la taille ait été réduite à 3.

(f)

Combiner ces deux modifications amplifierait l'erreur. Utiliser capacite au lieu de taille rendrait les accès aux éléments incorrects et la suppression de la mise à zéro laisserait des valeurs indésirables dans le tableau.

Cas de test :

- Tableau : [1, 2, 3]
- taille : 3
- capacite : 5
- Valeur à supprimer : 2

- Résultat attendu : [1, 3, 0], mais la suppression serait incorrectement exécutée, et les éléments non supprimés seraient déplacés en dehors de la portée réelle de la taille du tableau.

3)

Test 1 : [3,5,6,8] 4 4 7

Une implémentation qui échouerait à ce test pourrait ne pas insérer l'élément correctement à la position juste avant 8.

Implémentation défectueuse se trouve sur cette ligne :

for(i = taille; i > 0 && tab[i] > val; i--) devrait être tab[i-1]

Résultat attendu : [3, 5, 6, 7, 8] mais ici, il pourrait produire [3, 5, 6, 8, 7].

Test 2 : [2,4,5] 3 8 1

Une implémentation défectueuse pourrait ne pas insérer correctement 1 à la première position.

Implémentation défectueuse se trouve sur cette ligne :

for(i = taille; i > 0 && tab[i-1] >= val; i--) devrait être tab[i-1] > val

Cela insérerait 1 incorrectement, peut-être produisant [1, 1, 2, 4, 5].

4)

Pour tester la classe Tableau depuis l'extérieur, il serait utile d'avoir les observateurs suivants :

- int size() : pour obtenir la taille actuelle du tableau.
- int capacity() : pour obtenir la capacité maximale du tableau.
- int getElement(int index) : pour accéder à un élément à un index donné.
- boolean isFull() : pour vérifier si le tableau est plein.
- boolean isEmpty() : pour vérifier si le tableau est vide.

Ces méthodes permettraient de vérifier directement l'état interne du tableau après chaque opération.

5)

Avec ces méthodes, les éléments non observer sont :

Le Contenu exact du tableau

Si le tableau est remplie o si il est vide

Peuvent-ils être testés ? :

On peut tester indirectement l'état du tableau à travers des séries d'insertion/suppression et en vérifiant la taille avec size(), la capacité avec capacity(), et la présence d'un élément avec present(). Cependant, le test des éléments vides (comme l'absence de mise à zéro après suppression) serait difficile sans accès direct au tableau.

6)

Suppression du premier élément ([1, 2, 3, 4], supprimer 1)

1. Initialisation : créer un tableau avec une capacité de 4.
2. Insertion : insérer 1, 2, 3, 4 dans le tableau.
3. Suppression : appeler supprimer(1).
4. Vérification : vérifier que size() retourne 3, que present(1) retourne false et que present(2), present(3), present(4) retournent true.

Suppression dans un tableau vide ([], supprimer 5)

1. Initialisation : créer un tableau vide.
2. Suppression : appeler supprimer(5).
3. Vérification : vérifier que size() retourne 0, et que la fonction supprimer(5) retourne false.