TP6 : spécification et validation d'algorithmes de recherche

Vous devez spécifier et valider la méthode `recherche`. Cette méthode recherche l'indice de la cellule du tableau contenant un élément précis : `val.`

1. La méthode `rechercheLineaireTableauTrie` recherche uniquement dans un tableau trié. Elle est plus rapide que la méthode `rechercheLineaire` qui n'a pas de près-requis sur le tableau.
2. la méthode `recherche` appelle la méthode la plus adéquate ; à savoir `rechercheLineaireTableauTrie` si le tableau est trié sinon il appelle la méthode `rechercheLineaire.`
3. Pas de pré-condition pour les méthodes publiques.

Spécification informelle d'un algorithme de recherche dans un tableau d'entiers (`rechercheLineaire, rechercheLineaireTableauTrie, rechercheDichotomique, recherche`)

Si l'indice retourné par l'algorithme de recherche nommé `i,` est strictement inférieur à `zone.length` alors le contenu de `zone[i]` est égal à la valeur recherchée. Si l'indice retourné est `zone.length` alors la valeur recherchée n'est pas contenue dans le tableau.

Installation

1. Ouvrez une "fenêtre de commandes" Windows (onglet Démarrer, puis Exécuter "cmd"). Dans la fenêtre de commandes, tapez la commande suivante ou faites un copier/coller:
```
> set Path=%Path%;\\info\Bibliotheque\MF-dev\TP-EscJava\windows\escjava\bin
```

2. Placez-vous dans le répertoire `<REPERTOIRE_TP>\Exercices\4.tableauxTries`
```
> Z :                    > CD <REPERTOIRE_TP>\Exercices\4.tableauxTries
```

Pour utiliser ESC/Java, tapez la commande (Windows) suivante:
```
> escjava -loopSafe TableauTrie.java
```

Le répertoire `\\info\Bibliotheque\MF-dev\TP-EscJava\Doc` contient de la documentation sur ESC/Java.

Etape 1 : implémenter la méthode `estTrie`

La méthode retourne `true` si le tableau est trié dans l'ordre croissant sinon elle retourne `false`

1. Réécrivez le code de la méthode `estTrié`. Le code doit être conforme à la spécification :
```
/*@ ensures (\result == true ==>
      (\forall int l, j ; l >= 0 & l < j & j < zone.length
      ==> zone[l] <= zone[j]) );
*/
```
2. Validez votre code : aucune mise en garde ne doit être générée par `escjava`. N'oubliez pas de dé-commenter la spécification. C'est-à-dire que la ligne suivante doit être détruite
```
 /* la spécification suivante doit être vérifiée :
```

Etape 2 : spécifier de la méthode `rechercheLineaire`

1. Donnez la spécification la plus complète possible de `rechercheLineaire`. Pas de pré-condition (directive `requires`).
2. Donnez les invariants de boucle.
3. Validez votre code : aucune mise en garde ne doit être générée par `escjava`.

Etape 3 : spécifier et validation de la méthode `rechercheLineaireTableauTrie`

4. Reprenez la spécification de `rechercheLineaire`(directives `ensures`).
5. Ajoutez une/des pré-conditions (directive `requires`).
6. Donnez les invariants de boucle.
7. Validez votre code : aucune mise en garde ne doit être générée par `escjava`.

Etape 4 : spécifier et validation de la méthode `recherche`

8. Reprenez la spécification de `rechercheLineaire` (directives `ensures`).
9. Validez votre code : aucune mise en garde ne doit être générée par `escjava`.

```
public class TableauTrie {
   private /*@ spec_public */ /*@ non_null */ int[] zone;

     /* la spécification suivante doit être vérifiée :
     /*@ ensures (\result == true ==>
           (\forall int l, j ; l >= 0 & l < j & j < zone.length
           ==> zone[l] <= zone[j]) );
     */
     private boolean estTrie () {return true;}

     /*  algorithme : parcours d'un tableau non trié */
     public int rechercheLineaire(int val){
           int cpt = 0;
           while (cpt < zone.length) {
                 if (zone[cpt] == val) { return cpt;}
                 cpt++; }
           return cpt;
     }

     /* algorithme : parcours d'un tableau trié */
     private int rechercheLineaireTableauTrie(int val){
           int cpt = 0;
           while (cpt < zone.length) {
                 if (zone[cpt] == val) { return cpt; }
                 if (zone[cpt] > val) { return zone.length;}
                 cpt++; }
           return cpt;
     }

     /* On utilise l'algorithme le plus performant */
     public int recherche(int val){
            if (! estTrie()) return rechercheLineaire(val);
            return rechercheLineaireTableauTrie(val);
     }

     public TableauTrie() {zone = new int [0];}

}
```