Les tris (séance 1). Exercices

1. Ecrire une fonction booléenne qui, étant donné un tableau `tab` répond vrai si le tableau est (déjà) trié.
2. Comment peut-on modifier la plupart des algorithmes pour avoir un bon temps de calcul dans le meilleur cas ?
3. Ecrire une fonction de comparaison de tableaux de caractères qui étant donnés deux tableaux de caractères (type `String` en Java) , soient $s1$ et $s2$, renvoit -1 si $s1 < s2$, 0 si $s1 = s2$, et 1 si $s1 > s2$ (ordre lexicographique).
4. Proposer une version sans sentinelle de l'algorithme de tri par insertion, et déterminer le coût supplémentaire.
5. Proposer une version récursive de l'algorithme de tri par insertion.
6. **Champion borné** Écrire un programme qui recherche dans un tableau la plus petite valeur comprises entre deux bornes a et b données par l'utilisateur.
7. On suppose que l'on dispose d'un tableau de **chaînes de caractères**, de dimension N, correspondant à un ensemble de mots.

   ```
   String Tab[] ;
   tab = new String[N] ;
   tab = {"paul","marie","anne","marion","pierre"} ; // par exemple
   ```

   Ce tableau n'est pas trié. On veut produire un tableau trié sur l'envers des mots (on considère chaque mot « à l'envers », de la dernière lettre à la première)[1].

   (a) Écrire une methode `avant` qui prend en argument deux chaînes de caractère, et répond `vrai` si la première est avant la seconde, dans le sens défini précédemment (c'est-à-dire en prenant les mots à l'envers).

   ```
   public boolean avant(String s1, String s2) ;
   ```

   (b) Proposer un algorithme qui réalise le tri du tableau initial, en utilisant la methode `avant`.

   (c) Discuter et justifier l'algorithme que vous avez choisi.

[1] On utilise fréquemment ce genre de technique pour travailler sur les suffixes.