IF1: Introduction à l'informatique et à la programmation

Deuxième devoir sur table du groupe M1
2 décembre 2011 - Durée: 1h15

Documents non autorisés. Le barème est donné à titre indicatif.
La lisibilité du code Java sera prise en compte dans l'évaluation.

Exercice 1 (**10 points**) Étant donné un tableau d'entiers `t` de longueur n, on dira qu'une *marche montante dans* `t` est un entier i, $0 \leq i < n-1$, tel que `t[i]<t[i+1]`. De même, une *marche descendante dans* `t` est un entier i, $0 \leq i < n-1$, tel que `t[i]>t[i+1]`.

1. Écrire une méthode `niveauFinal` qui prend comme argument un tableau d'entiers `t` et qui renvoie l'entier égale au nombre de marches montantes dans `t` moins le nombre de marches descendantes dans `t`. Par exemple, l'appel `niveauFinal({1,3,5,7,6,4})` doit renvoyer `1`.

2. Écrire une méthode `estTrie` qui prend comme argument un tableau d'entiers et qui vérifie si ce tableau est trié pour l'ordre strictement croissant[1], en utilisant la méthode `niveauFinal` . Le résultat de `estTrie` est de type `boolean`.

3. Quand une marche descendante dans `t` suit immédiatement une marche montante dans `t`, ou vice-versa, on dit que `t` a un *pic*. Écrire une méthode `nombreDePics` qui prend comme argument un tableau d'entiers et qui renvoie son nombre de pics. Par exemple, l'appel `nombreDePics({1,2,1,2,3,2})` doit renvoyer `3`.

Exercice 2 (**10 points**)

Les chaînes de caractères peuvent être ordonnées par l'ordre alphabétique, dont la définition précise est rappelée ci-dessous:

Soient v et w deux mots sur l'alphabet $\{a, b, c, \ldots, x, y, z\}$. On dit que v *est plus petit que* w *pour l'ordre alphabétique* s'il existe un entier $n \geq 0$ tel que:

- v et w sont identiques jusqu'à leur n-ième caractère inclus, et:
- – Soit la longueur de v est n;
  – Soit le $(n+1)$-ième caractère de v précède le $(n+1)$-ième caractère de w dans l'ordre $a < b < c < \ldots < x < y < z$.

Rappels: si `s` est un `String`, l'appel `s.length()` renvoie la longueur de `s`; si `t` est un deuxième `String`, l'appel `s.equals(t)` renvoie `true` si `s` et `t` sont égaux, `false` sinon. La méthode `char charAt(int n)` renvoie le n-ième caractère d'un `String` comme dans les exemples suivants:

`"toto".charAt(0)` renvoie le caractère `t`, et `"toto".charAt(3)` renvoie le caractère `o`.

1. Écrire une méthode `sontOrdonnes` qui prend comme arguments deux `String` et qui vérifie si le premier est plus petit que le deuxième, selon la définition ci-dessus.

2. On suppose d'avoir à disposition un tableau de mots:

   `String[] dictionnaire={"abaissant","abandon",...,"zozoter","zygote"}`

[1] C'est à dire que tout élément de `t` est strictement plus petit que l'élément suivant.

Écrire une méthode `boolean chercheDansLeDictionnaire(String s)` qui vérifie si le mot `s` est dans le dictionnaire. La recherche commence au début du dictionnaire, mot par mot, et s'arrête dès que `s` dépasse un élément du dictionnaire, pour l'ordre alphabétique.

Le dictionnaire n'est pas un paramètre de la méthode, mais une variable globale visible à l'intérieur de `chercheDansLeDictionnaire`.

3. Esquisser l'implantation d'une fonction de recherche d'un mot dans le dictionnaire qui soit plus efficace que celle implementée par la méthode `chercheDansLeDictionnaire` (l'idée étant de remplacer la recherche *séquentielle* par une recherche *dichotomique*, similaire à celle qu'on utilise quand on consulte un dictionnaire).