Licence de Mathématiques et Informatique
Semestre 2

Université de Perpignan Via Domitia

Algorithmique

Contrôle continu de mars 2017

NOM : **Prénom :** **Groupe TD :**

Durée : 1 heure. Aucun document ni téléphone autorisé.

Modalités : Répondre uniquement dans les cadres prévus à cet effet. Un barème sur 30 est indiqué à droite de chaque question. La note finale sur 20 est proportionnellement déduite de ce barème. La qualité de la rédaction sera prise en compte dans la notation. Les questions avec un symbole ($\star$) sont de niveau Objectif 20.

Total : **/30** **/20**

Exercice 1. (18 points) **/18**

1. Ecrire l'en-tête d'une fonction `nbOcc( )` qui compte le nombre d'occurrences d'une valeur donnée dans un ensemble de n valeurs. On se place dans le cas où l'ensemble de valeurs est un tableau de caractères `s[n]`, *i.e.* une chaîne de caractères de longueur n. **/1**

2. Ecrire la fonction `nbOcc( )` qui applique un algorithme itératif.
   Ce n'est pas la peine de ré-écrire la description de la fonction (le commentaire *docstring* d'en-tête). **/3**

3. Utiliser `nbOcc( )` pour compter les nombres d'occurrences suivantes qui seront *ensuite* affichées.
   — `n0` : nombre de `a` dans `licence` ?
   — `n1` : nombre de `i` dans `licence` ?
   — `n2` : nombre de `c` dans `licence` ?

/2

4. Justifier le choix de la structure de contrôle utilisée dans `nbOcc( )`. **/1**

5. En déduire la complexité asymptotique, en temps, de `nbOcc( )`. Préciser la mesure et le paramètre de complexité associés. **/2**

6. (⋆) Ecrire `nbOccRec( )`, version récursive de `nbOcc( )`. **/4**

7. Utiliser `nbOccRec( )` pour compter les nombres d'occurrences suivantes qui seront ensuite affichées.
   — `n0` : nombre de `a` dans `licence` ?
   — `n1` : nombre de `i` dans `licence` ?
   — `n2` : nombre de `c` dans `licence` ?

/2

8. ($\star$) En notant `nbOccRec` par `f` (pour faire plus court), exhiber l'évolution de la pile des appels à `f` pour les décomptes suivants.

   (a) nombre de `e` dans `ete`. /2

   (b) nombre de `a` dans `ete`. /1

Exercice 2. (12 points) /12

1. Écrire la fonction itérative `minIt( )` qui calcule le minimum des valeurs d'un tableau d'entiers de longueur n arbitraire : $t = [a_0, a_1, \cdots, a_{n-1}]$. /3

2. Quelle est la complexité (en temps) asymptotique de cette solution itérative ? **/1**

3. ($\star$) On dispose maintenant d'une fonction `min(a,b)` qui retourne le minimum de deux entiers de signe quelconque `a` et `b`. Identifier une propriété qui permet de calculer de façon récursive la valeur minimale (de signe quelconque) présente dans le tableau t. Le cas terminal sera explicité et la fonction `min(a,b)` sera utilisée. **/3**

4. ($\star$) Écrire la fonction récursive `minRec( )` qui calcule le minimum des valeurs d'un tableau d'entiers `t` de longueur n arbitraire. Cette version utilisera la fonction `min(a,b)`. **/3**

5. ($\star$) Quelle est la complexité (en temps) asymptotique de cette solution ? Qu'en penser ? **/2**