Département Informatique – IUT A Lyon 1 Janvier 2016

Nom : ..... Groupe : .....

Algorithmique – Devoir surveillé n°2

Durée : 1h30 – aucun document autorisé
Le barème est donné à titre indicatif
Un point est réservé à la présentation

Exercice 1 (4 points)

On considère l'algorithme suivant :

```
 1  def mystère(tab):
 2    x = tab[0]
 3    y = tab[0]
 4    i = 1
 5    while i < len(tab):
 6        if tab[i] < x:
 7            x = tab[i]
 8        if tab[i] > y:
 9            y = tab[i]
10        i = i+1
11    i = 0
12    while i < len(tab):
13        tab[i] = (tab[i]-x)/(y-x)
14        i = i+1
```

Votre travail :

a) Écrivez la trace de cet algorithme pour le tableau [11, 14, 10, 12, 11].

b) Donnez la spécification (entrées, sorties, pré-conditions, post-conditions) de cet algorithme.

Exercice 2 (4 points)

On vous demande d'écrire une fonction **récursive** qui teste si une portion (délimitée par deux indices) d'un tableau de nombres flottants est triée. Elle renverra True si la condition est vérifiée et False dans le cas contraire.

Par exemple, pour le tableau suivant :

	0	1	2	3	4	5	6	7	8	9
t =	2.4	5.6	1.2	2.3	3.4	4.5	5.6	6.7	5.4	4.3

test_trié_croissant(*t*, 2, 7) retournera True, mais *test_trié_croissant*(*t*, 2, 8) retournera False.

Votre travail : écrire la fonction *test_trie_croissant* spécifiée ci-dessous.

```
def test_trié_croissant (tab, imin, imax) :
    """
    :entrée tab: tableau de float
    :entrée imin: int
    :entrée imax: int
    :pré-cond : 0 ≤ imin ≤ imax < len(tab)
    :sortie trié: bool
    :post-cond: trié==True si tab est trié dans l'ordre croissant entre les indices imin et imax (inclus),
            et trié==False dans le cas contraire.
```

"""

Exercice 3 (5 points)

On considère un tableau de flottants *t* contenant beaucoup de zéros. Afin d'économiser de la mémoire, on souhaite représenter ces données à l'aide de deux tableau *tind* et *tval*, contenant les mêmes informations que *t*, mais structurés différemment. Plus précisément, si *t* contient *n* valeurs non nulles, *n* sera la longueur des tableaux *tind* et *tval*, et pour chacune de ces valeurs v_i (avec $0 \leq i < n$), *tval*[*i*] aura pour valeur v_i, et *tind*[*i*] aura pour valeur l'indice de v_i dans *t*. Enfin, l'ordre d'apparition des valeurs dans *tval* doit être le même que dans *t*, ce qui revient à dire que les indices stockés dans *tind* doivent être dans l'ordre croissant.

Par exemple, avec le tableau *t* suivant :

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
t =	0	0	5.2	0	0	0	6.7	1.3	0	8	0	0	0	0	0	9.1	3.4	0	0

On obtiendra le résultat suivant :

	0	1	2	3	4	5
tind =	2	6	7	9	15	16

	0	1	2	3	4	5
tval =	5.2	6.7	1.3	8	9.1	3.4

Votre travail : écrire la fonction *compresse_tableau* spécifiée ci-dessous.

```
def compresse_tableau (t) :
    """
    :entrée t: tableau de float
    :sortie tind: tableau d'int
    :sortie tval: tableau de float
    :post-cond: len(tind) = len(tval) = nombre de valeurs non nulles dans t
        ∀i ∈ [0;len(t)[, si t[i] ≠ 0, i ∈ tind et t[i] ∈ tval   (tind et tval contienne la même information que t)
        ∀i ∈ [0;len(tind)[, tval[i] = t[tind[i]]                (tind contient l'indice de v_i, tval contient sa valeur)
        ∀i ∈ ]0;len(tind)[, tind[i] = t[tind[i]]                (les indices dans tind sont dans l'ordre croissant)
    """
```

Exercice 4 (6 points)

(cet exercice peut-être traité indépendamment de l'exercice précédent)

On suppose maintenant qu'on dispose des deux tableaux *tind* et *tval* produits dans l'exercice précédent, et qu'on a donc **effacé le tableau *t* pour économiser de la mémoire**. On souhaite écrire un algorithme retrouvant la valeur qui était stockée dans *t* à un indice donné *it*, en recherchant *it* dans le tableau *tind*. Si *it* est présent dans *tind*, alors la valeur correspondante dans *tval* est la valeur recherchée ; dans le cas contraire, c'est que *t*[*it*] valait zéro (ou que len(*t*) était inférieure à *it*, auquel cas on retournera zéro également).

On exploitera le fait que *tind* est trié, en effectuant une recherche *dichotomique*.

Votre travail : écrire la fonction *extraire_valeur* spécifiée ci-dessous.

```
def extraire_valeur (tind, tval, it) :
    """
    :entrée tind: tableau d'int
    :entrée tval: tableau de float
    :entrée it: int
    :sortie val: float
    :pré-cond: tind et tval représentent un tableau compressé au sens de l'exercice 1, it ≥ 0
    :post-cond: val est la valeur à l'indice it du tableau représenté par tind et tval
            ou 0 si it n'est pas un indice de ce tableau
    """
```