info201 : algorithmique et programmation
Examen
CORRECTION

Pierre Hyvernat
Ilham Alloui
UFR Sciences Fondamentales et Appliquées, université de Savoie
Pierre.Hyvernat@univ-smb.fr
Ilham.Alloui@univ-smb.fr

Documents et calculatrices interdits.

Durée : 1h30.

Un barème provisoire est donné dans la marge, un point négatif est réservé pour la présentation.

Rappel : vous pouvez utiliser les fonctions demandées pour écrire celles des questions suivantes, *même si vous ne les avez pas écrites.*

Partie 1 : tableaux, chaines et boucles

(2) *Question 1.* Écrivez une fonction `compte_pos(T)` qui compte le nombre d'éléments strictement positifs dans un tableau de nombres.

Solution :
```
def compte_pos(T):
    r = 0
    for e in T:
        if e > 0:
            r = r + 1
    return r
```

(2) *Question 2.* Écrivez une fonction `est_trie(T)` qui vérifie qu'un tableau de nombres est trié dans l'ordre croissant.

Remarque : 1 point du barème est réservé au fait que votre fonction s'arrête dès que possible.

Solution :
```
def est_trie(T):
    r = True
    i = 1
    while i<len(T) and r:
        if T[i-1] > T[i]:
            r = False
        i = i+1
    return r
```

(2) *Question 3.* Écrivez une fonction `mot_titre(s)` qui vérifie qu'une chaine de caractères est un mot commençant par une majuscule :
- le premier caractère est une lettre en majuscule,
- tous les autres caractères sont des caractères alphabétiques en minuscules.

Remarque : 1 point du barème est réservé au fait que votre fonction s'arrête dès que possible.

```
>>> mot_titre("Bonjour")
True
>>> mot_titre("Bonjour.")
False
```

Pour ceci, vous pouvez utiliser les méthodes suivantes sur les chaines de caractères :
- `s.islower()` / `s.isupper()` pour vérifier si une chaine est en minuscules / en majuscules
- `s.isalpha()` pour vérifier si une chaine est constituée de caractères alphabétiques.

Les autres méthodes sur les chaines ne sont pas autorisées.

Solution :

```
def mot_maj(s):
    if not s.isalpha():
        return False
    if not s[0].isupper():
        return False
    i = 1
    r = True
    while i < len(s) and r:
        if not s[i].islower():
            r = False
        i = i+1
    return r
```

(3) Question 4. Écrivez une fonction `contient_maj(T)` qui vérifie si chacune des chaines d'une liste contient au moins une majuscule.

Remarque : 1 point du barème est réservé au fait que votre fonction s'arrête dès que possible.

Solution :

```
def contient_maj(T):
    r = True
    i = 0
    while i<len(T) and r:
        maj = False
        j = 0
        while j<len(T[i]) and not maj:
            if T[i][j].isupper():
                r = True
            j = j+1
        r = maj
        i = i+1
    return r
```

(3) Question 5. Écrivez une fonction `compte_dieses(s)` qui compte le nombre maximum de "`#`" consécutifs dans une chaine.

```
>>> compte_dieses("abcd")
0
>>> compte_dieses("a#bc###d#")
3
```

Solution :

```
def compte_dieses(s):
    r = 0
    k = 0
    for i in range(len(s)):
        if s[i] == '#':
```

```
            k = k+1
            if k > r:
                r = k
            else:
                k = 0
    return r
```

ou bien

```
def compte_dieses(s):
    r = 0
    i = 0
    while i<len(s):
        k = 0
        while i+k<len(s) and s[i+k] == '#':
            k = k+1
        if k > r:
            r = k
        i = i+k+1
    return r
```

Partie 2 : Dictionnaires

(2) *Question 1.* Écrivez une fonction qui calcule la moyenne des valeurs d'un dictionnaire.
La seule méthode autorisée sur les dictionnaires est la méthode `D.keys()`.

Solution :

```
def moyenne_dico(D):
    m = 0
    for k in D.keys():    # ou    for k in D:
        m = m + D[i]
    return m/len(D)
```

(2) *Question 2.* On dispose d'un dictionnaire `anglais` qui associe à des mots français, leur traduction en anglais :

```
{ "ordinateur": "computer", "clavier": "keyboard", "souris": "mouse", ... }
```

Écrivez une fonction `inverse` qui calcule le dictionnaire "à l'envers", qui associe aux mots anglais, leurs traduction en français :

```
{ "computer": "ordinateur", "keyboard": "clavier", "mouse": "souris", ... }
```

Solution :

```
def inverse(D):
    D2 = {}
    for k in D.keys():
        D2[D[k]] = k
    return D2
```

Partie 3 : problème : fractions continues

Une fraction continue est une expression de la forme

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ldots + \cfrac{1}{a_n}}}}}$$

Par exemple, pour $a_0 = 2$, $a_1 = 3$ et $a_2 = 4$, on obtient

$$2 + \frac{1}{3 + \frac{1}{4}} \quad = \quad 2 + \frac{1}{\frac{13}{4}} \quad = \quad 2 + \frac{4}{13} \quad = \quad \frac{30}{13} \quad \approx \quad 2.307692307...$$

(2) *Question 1.* En constatant que l'on fait le calcul à partir de la fin ("$3 + 1/4 = 13/4$ dans l'exemple précédent), complétez la fonction `fraction_continue` qui prend en argument un tableau de nombres entiers : `[a1, a2, ..., an]` et qui renvoie le nombre correspondant.

```
>>> fraction_continue([2,3,4])
2.3076923076923075
```

Voici le programme à compléter :

```
def fraction_continue(A):
    # on initialise le résultat avec la dernière valeur
    r = A[-1]
    # on parcourt toutes les autres valeurs, en commençant par la fin
    for i in range(len(A)-2,-1,-1):
        ...
    return r
```

Solution :

```
def fraction_continue(A):
    # on initialise le résultat avec la dernière valeur
    r = A[-1]
    # on parcourt toutes les autres valeurs, en commençant par la fin
    for i in range(len(A)-2,-1,-1):
        r = A[i] + 1/r
    return r
```

(2) *Question 2.* L'opération inverse permet de retrouver une fraction continue à partir d'un nombre à virgule.

Par exemple, 5.4321 est de la forme $5 + 1/x$ car sa partie entière est 5. Pour continuer, il faut calculer x, qui est égal $1/0.4321$ et recommencer l'opération :

```
x = 5.4321        =>      a0 = 5,    x = 1/0.4321      = 2.314279...
x = 2.314279...   =>      a1 = 2,    x = 1/0.314279... = 3.181885...
x = 3.181885...   =>      a2 = 3,    x = 1/0.181885...
```

Sur chaque ligne, on calcule un entier `an` et la nouvelle valeur de `x`.

Écrivez une fonction `fraction_continue_inverse(x, n)` qui calcule le tableau des nombres `[a1, a2, ...]` jusqu'à ce que l'une des conditions suivantes soit vraie :

- le nombre `x` est un entier, auquel cas, le nombre `ai` correspondant est égal à `x` et la boucle s'arrête,
- on a calculé `n` nombres `a1, ..., an`.

Par exemple :

```
>>> fraction_continue_inverse(5.4321, 4)
[5, 2, 3, 5]
>>> fraction_continue_inverse(2.5, 10)
[2, 2]
```

Solution :

```
def fraction_continue_inverse(x, n):
    A = []
    i = 0
    fini = False
    while not fini and i<n:
        a = int(x)
```

```
    A.append(a)
    if a == x:
        fini = True
    else:
        x = 1/(x-a)
    i = i+1
return A
```