INF1 – P23

Examen Médian

Durée : 1 heure 30

Calculatrices, téléphones, traducteurs et ordinateurs interdits.

Attention : chaque partie doit être rédigée sur une copie séparée

N.B. : on s'attachera à fournir tout commentaire utile et à écrire de façon claire et lisible.

1ère partie : Listes et tableaux *(6 points)*

Soit l'algorithme ci-dessous, qui prend en entrée un tableau, tab, de n nombres entiers :

```
Entrée : tab, n
j <- 0
k <- 0
y <- 0
z <- 0
pour i allant de 1 à n faire
     si tab[i] % 2 = 0 alors
          j <- j + 1
          tab1[j] <- tab[i]
          y <- y + tab[i]
     sinon
          k <- k + 1
          tab2[k] <- tab[i]
          z <- z + tab[i]
     fin_si
fin_pour
si j > 0 alors
     y <- y / j
fin_si
si k > 0 alors
     z <- z / k
fin_si
afficher("point 1 : ", y)
afficher("point 2 : ", z)
si y>z alors
     afficher("point 3-1 : Voici les nombres de la série 1 :"
et rester à la ligne)
     pour i allant de 1 à j faire
          afficher(tab1[i] et rester à la ligne)
     fin_pour
sinon
     afficher("point 3-2 : Voici les nombres de la série 2 :"
et rester à la ligne)
     pour i allant de 1 à k faire
          afficher(tab2[i] et rester à la ligne)
     fin_pour
fin_si
```

1. Considérer les entrées suivantes :
   Entrée 1 : `n = 10 ; tab = [1 3 5 7 9 2 6 3 8 4]`
   Entrée 2 : `n = 5 ; tab = [3 2 6 11 7]`
   Pour chacune de ces entrées, quel affichage on aura aux points d'affichage : Point 1, Point 2, Point 3-1 et Point 3-2 ?

```
'''
Pour l'entrée 1
    point 1 :  5.0
    point 2 :  4.6
    point 3-1 : Voici les nombres de la série 1 : 2 6 8 4
Pour l'entrée 2
    point 1 :  4.0
    point 2 :  7.0
    point 3-2 : Voici les nombres de la série 2 : 3 11 7
'''
```

2. Décrire ce que fait cet algorithme.

```
'''Cet algorithme calcule la moyenne des nombres pairs et impairs
   présents dans le tableau, affiche les valeurs de ces moyennes,
   ensuite si la moyenne des nombres pairs est supérieure à la
   moyenne des nombres impairs l'algorithme affiche la liste des
   nombres pairs, sinon il affiche la liste des nombres impairs.'''
```

3. Ecrire le programme python correspondant.

```
#n= 10
#tab = [1, 3, 5, 7, 9, 2, 6, 3, 8, 4]
n = 5
tab = [3, 2, 6, 11, 7]
j = 0
k = 0
y = 0
z = 0
tab1 = []
tab2 = []
for i in range(0,n) :
    if tab[i] % 2 == 0 :
        j = j + 1
        tab1.append(tab[i])
        y = y + tab[i]
    else :
        k = k + 1
        tab2.append(tab[i])
        z = z + tab[i]
if j > 0 :
    y = y / j
if k > 0 :
    z = z / k
print("point 1 : ", y)
print("point 2 : ", z)
if y > z :
    print("point 3-1 : Voici les nombres de la série 1 :", end=" ")
    for i in range(0,j):
        print(tab1[i], end = " ")
else:
```

```
    print("point 3-2 : Voici les nombres de la série 2 :", end=" ")
    for i in range(0,k):
        print(tab2[i], end = " ")
```

---------- ***{ prendre une nouvelle copie }*** ----------

2ème partie : Structures itératives *(6 points)*

1. Qu'est-ce qu'une structure itérative en algorithmique ?
2. Donner les trois structures itératives identifiées en algorithmique et préciser leurs différences ainsi que les conditions idéales d'utilisation.
3. Pour chacune des trois structures identifiées en algorithmique, donner sa mise en œuvre en python.

   Écrire un programme en python qui affiche les 20 premiers termes de la table de multiplication par 7, en signalant au passage (à l'aide d'un astérisque) ceux qui sont des multiples de 3. Justifiez le choix de la structure utilisée.
   Exemple : 7 14 21 * 28 35 42 * 49 56 63 * 70 77 84 * 91 98 105 * 112 119 126 * 133 140

```
for i in range(1, 21):
    t = i * 7
    print(t, end = '  ')
    if t % 3 == 0:
        print('*', end = '  ')
```

---------- ***{ prendre une nouvelle copie }*** ----------

3ème partie : Chaines de caractères *(8 points)*

Soit les chaines de caractères suivantes :

```
S1 = "123456789"
S2 = "abcdefghi"
S3 = "Université de Technologie de Compiègne"
S4 = "portez ce vieux whisky au juge blond qui fume"
```

1. En utilisant le slicing Python extraire la chaine `"5678"` de `s1`.

```
    s1[4:8]
```

2. En utilisant le slicing Python, comment extraire de la chaine `s2` une lettre sur deux à l'envers et obtenir `"igeca"` ?

```
    s1[4:8]
```

3. Ecrire le programme Python qui permet de créer une chaine de caractère `s5` qui mélange des chaines de caractères de même longueur. En mélangeant `s1` et `s2`, on obtient : `"1a2b3c4d5e6f7g8h9i"`.

```
s5 = ""
for i in range (len(s1)):
    s5 = s5 + s1[i] + s2[i]
print(s5)
```

4. Modifier le programme précédent pour mélanger des chaines de longueurs différentes. En mélangeant `s1` et `s3`, on obtient ; "1U2n3i4v5e6r7s8i9té de Technologie de

Compiègne" .

```
s5 = ""
if len(s1) < len(s3) :
    min = len(s1)
    s6 = s3[len(s1):]
else :
    min = len(s3)
    s6 = s1[len(s3):]
for i in range (min) :
    s5 = s5 + s1[i] + s3[i]
s5 = s5 + s6
print(s5)
```

5. Ecrire un algorithme ou programme Python qui affiche les lettres majuscules de la chaine `s3`, ici `"UTC"`

```
acronyme = ""
for c in s3 :
    if c >= 'A' and c <= 'Z' :
        acronyme += c
print(acronyme)
```

6. Un pangramme est une phrase comportant au moins une fois chaque lettre de l'alphabet. Par exemple, `s4` est un pangramme. Ecrire un algorithme ou programme Python qui affiche si une chaine de caractère est un pangramme ou non. On considère que toutes les lettes sont des minuscules.

```
pangramme = True
alphabet = "abcdefghijklmnopqrstuvwxyz"
i = 0
while i < len(alphabet) and pangramme :
    if not alphabet[i] in s4 :
        pangramme = False
    else :
        i += 1
if pangramme :
    print("La chaine est un pangramme")
else :
    print("La chaine n'est pas un pangramme")
```