INF1 – P22

Examen Médian

Durée : 1 heure 30

.

Document autorisé : une feuille de notes A4 recto-verso
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 : alternatives *(6 points)*

1. Écrire un programme permettant de saisir le prix de fabrication et le prix de vente d'un produit et de déterminer s'il y a profit ou perte.
   Si le prix de fabrication est supérieur au prix de vente, il y a perte et sinon profit.

   Améliorer ce programme pour permettre à l'utilisateur de saisir plusieurs prix de produits. La saisie s'arrêtera lorsque l'utilisateur entrera 0 pour le prix de fabrication.

```
pf=float(input("Prix de fabrication : "))
while pf != 0:
    pv=float(input("Prix de vente : "))

    if (pv > pf):
        # calculer profit */
        montant = pv - pf
        print("Profit = ", montant)
    elif (pf > pv):
         # calculer la perte */
        montant = pf - pv
        print("Perte = ", montant)
    else:
        # Ni profit ni perte */
        print("Ni profit ni perte.")
    pf=float(input("Prix de fabrication : "))
```

2. Écrire un programme permettant d'afficher le nombre de jours d'un mois en fonction de son numéro (entre 1 et12).

   N.B. : si le numéro de mois est 2 (mois de février), il faudra demander à l'utilisateur le numéro de l'année afin de déterminer si l'année est bissextile et préciser alors 28 ou 29 jours.

   On rappelle qu'une année est bissextile si elle est divisible par 4. Toutefois, les années divisibles par 100 ne sont pas bissextiles, à l'exception des années divisibles par 400 qui, elles, le sont.

   Le nombre total de jours dans un mois est donné par le tableau ci-dessous.

Mois	Nombre de jours
janvier, mars, mai, juillet, août, octobre, décembre	31 jours

Mois	Nombre de jours
février	28/29 jours
avril, juin, septembre, novembre	30 jours

```
mois=int(input("Saisir le numéro du mois : "))
if mois in [1,3,5,7,8,10,12]:
    print("31 jours")
elif (mois == 2):
    annee=int(input("Saisir le numéro de l'année : "))
    if(((annee % 4 == 0) and (annee % 100 !=0)) or (annee % 400==0)) :
       print("29 jours")
    else :
       print("28 jours")
elif mois in [4,6,9,11]:
    print("30 jours")
else:
       print("Entrée invalide! Veuillez saisir le numéro du mois entre (1-
12).")
```

ou

```
mois=int(input("Saisir le numéro du mois : "))
if (mois == 1)or(mois == 3) or(mois == 5) or (mois == 7)or (mois ==
8)or(mois == 10) or(mois == 12):
    print("31 jours")
elif (mois == 2):
    annee=int(input("Saisir le numéro de l'année : "))
    if(((annee % 4 == 0) and (annee % 100 !=0)) or (annee % 400==0)) :
       print("29 jours")
    else :
       print("28 jours")
elif (mois == 4) or (mois==6) or (mois==9) or (mois ==11) :
    print("30 jours")
else:
        print("Entrée invalide! Veuillez saisir le numéro du mois entre (1-
12).")
```

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

2ème partie : séquences d'ADN *(7 points)*

Une séquence d'ADN (acide désoxyribonucléique) est constituée de 4 éléments : Adénine (A), Cytosine(C), Guanine (G) et Thymine (T).

On appellera « chaîne » d'ADN une chaîne de caractères constituée de caractères A, C, G ou T, et « motif » une sous-chaîne d'une chaîne d'ADN.

Une chaîne ou un motif sont « valides » s'ils ne sont pas vides et s'ils sont formés exclusivement d'une combinaison arbitraire de ' A', ' T', 'G' ou 'C'.

1- Écrire un algorithme pour une fonction `valide(ch)` qui renvoie `vrai` si la chaîne `ch` passée en paramètre est valide et `faux` sinon. Écrire ensuite la fonction Python correspondante.

```
Fonction valide(ch)
Variables
     ret : booléen
     c : caractère
     i : entier
Début
     si ch est vide alors
          ret <- faux
     sinon
         ret <- vrai
      i <- 1
      tant que ret = vrai et i <= longueur(ch)
             si ch[i] n'appartient pas à  {a, t, c, g} alors
                  ret <- faux
             i <- i + 1
      ftq
     retourner ret
Fin
```

En python

```
# 1ère solution
def valide(seq):
    ret = seq != ''   # ret initialisé à True si seq n'est pas vide, False
sinon
    i=0
    while ret and i < len(seq):
        ret = seq[i] in "atgc"
        i=i+1
    return ret

# 2ème solution
def valide(seq) :
    ret = seq != ''
    for c in seq :
        ret = ret and c in "atgc"
    return ret
```

2- Écrire une fonction python `compte(ch)` qui renvoie le nombre de caractères de la chaîne `ch` si elle est valide et `-1` sinon.

```
def compte(ch) :
    l=len(ch)
    if valide(ch):
        return l
    else:
        return -1
```

3- Écrire une fonction `saisie()` qui effectue une saisie valide et renvoie la valeur saisie sous forme d'une chaîne de caractères.

```
def saisie() :
    seq = input()
    while not valide(seq) :
        print(f'{seq} ne peut contenir que les lettre "a", "t", "g" et "c"
et ne doit pas
                être vide')
```

```
        seq = input()
    return seq
```

4- Écrire une fonction `proportion(motif, ch)` qui reçoit deux arguments, un motif et une chaîne et et qui retourne la proportion d'occurrences du motif dans la chaîne (c'est-à-dire le nombre de fois où le motif apparaît dans la chaîne).

```
def proportion(motif, ch) :
    return 100 * ch.count(motif) * len(motif) / len(ch)
```

On pouvait aussi réécrire une fonction count

5- Écrire le programme principal qui appelle la fonction `saisie()` pour la chaîne et pour le motif et qui affiche un résultat analogue à l'exemple suivant :

```
Entrez la chaine ADN: atgcaa
Entrez la chaine sequence: ca
Il y a 33.333333 % de ca dans votre chaine d ADN qui contient 6 caracteres
```

```
#                              Programme                              principal
============================================================
print("séquence d'ADN : ")
adn = saisie()
print("motif : ")
motif = saisie()
print(f"Il y a {proportion(motif, adn)} % de {motif} dans votre chaine
d'ADN qui contient
        {compte(adn)} caracteres")
```

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

3ème partie : tableaux *(7 points)*

1. Pair ou impair ?

Écrire un algorithme permettant de remplir un tableau avec une série de `n` nombres entiers entrés au clavier, puis de calculer la moyenne des nombres pairs de ce tableau. Le nombre `n` sera lui-aussi saisi au clavier.

L'algorithme affichera "Ce tableau est pair !", si tous les nombres impairs sont inférieurs à cette moyenne et "Ce tableau est impair !" sinon.

```
Variables
     n, i, nombre, nbPairs : entiers
     moyPairs : réel
     tab : tableau d'entiers

Début
afficher " Combien de nombres ? "
lire n
Initialiser le tableau à vide
moyPairs <- 0
nbPairs <- 0
pour i allant de 1 à n
     lire nombre
     ajouter nombre à tab
     si nombre modulo 2 = 0
```

```
            moyPairs <- moyPairs + 1
            nbPairs <- nbPairs +1

si nbPairs > 0 alors
      moyPairs <- moyPairs / nbPairs
      estPair <- vrai
      i <- 0
      tant que estPair = vrai et i < = n
            si tab[i] modulo 2 != 0 et tab[i] > moyPairs alors
                  estPair <- faux
             sinon
                   i <- i +1
sinon
      estPair <- faux

si estPair alors
      afficher « Le tableau est pair »
sinon
      afficher « Le tableau est impair »
Fin
```

Écrire le programme Python correspondant.

```
print(" ------ Question 1 : pair ou impair ? -----")
n = int(input("Saisir le nombre d'éléments : "))
moy_pairs = 0
tab = []
print("Entrez les éléments (non nuls) du tableau :")
nb_pairs = 0
# remplissage du tableau et somme des nombres pairs
for i in range(n):
    nombre = int(input(f'nombre {i} : '))
    tab.append(nombre)
    if tab[i] % 2 == 0:
        moy_pairs += tab[i]
        nb_pairs += 1

# le tableau est-il pair ou impair ?
if nb_pairs > 0 :
    moy_pairs /= nb_pairs
    est_pair = True
    i = 0
    while est_pair and  i < N :
        if tab[i] % 2 != 0  and  tab[i] > moy_pairs :
            est_pair = False
        else :
            i = i+1
else :
    est_pair = False

#affichage du message
print("Moyenne des nombres pairs : ", moy_pairs)
if est_pair :
    print("Ce tableau est pair !")
else :
    print("Ce tableau est impair !")
```

2. Les pairs d'abord !

a) Écrire une fonction python `classif_pair_impair(tab)` qui prend en entrée un tableau `tab` de nombres entiers et classe le tableau de manière à ce que tous les nombres pairs se retrouvent au début du tableau et les nombres impairs à la fin. L'ordre des nombres

pairs dans le tableau sera gardé. Le tableau sera modifié sans avoir recours à un autre tableau.

```
print(" ------ Question 2 : les pairs d'abords -----")
def classif_pair_impair(tab):
      i = 0  # indice du premier élément impair du tableau
      j = 0  # indice de l'élément courant
      # recherche de l'élément pair suivant
      while j < len(tab) :
          if tab[j] % 2 == 0 :
              # permutation de tab[i] et tab[j] et décalage des éléments
impairs
              temp = tab[j]
              for k in range(j,i,-1) :
                  tab[k] = tab[k-1]
              tab[i] = temp
              # on avance les 2 indices
              i += 1
              j += 1
          else :
              #  l'élément courant du tableau est impair : on avance
seulement j
              j += 1
      return tab
```

b) Écrire un programme principal qui fait appel à la fonction `classif_pair_impair(tab)` et affiche le résultat comme dans l'exemple suivant :

Exemple :

Tableau transmis à la fonction : [24, 13, 17, 16, 2, 5, 7, 42]

Tableau après classification pairs et impairs : [24, 16, 2, 42, 13, 17, 5, 7]

```
# Saisie de la taille de la série
n = int(input("Saisir le nombre d'éléments : "))
# Saisie des éléments du tableau
tab = []
for i in range(n):
    tab.append(int(input(f"Entrez l'élement {i+1} : ")))

print(classif_pair_impair(tab))
```

c) La fonction `classif_pair_impair` modifie le tableau passé en paramètre, ce qui n'est pas une bonne pratique en général. En utilisant cette fois plusieurs tableaux, écrire une fonction `classif_pair_impair_2` qui retourne un tableau classé comme précédemment mais sans modifier le tableau passé en paramètre.

```
def classif_pair_impair_2(tab):
    pairs = []
    impairs = []

    for elt in tab:
        if elt % 2 == 0 :
            pairs.append(elt)
        else :
            impairs.append(elt)
        tab = pairs + impairs
        return tab
```