

Devoir de contrôle n°2

Matière : INFORMATIQUE

Classe : 1^{ère} Année MP – PT – PC Durée : 1h

*Toutes les réponses doivent être rédigées en Python.
L'indentation dans les instructions Python sera prise en considération dans le barème*

Exercice 1 : (8 points)

Soit **u** la suite récurrente d'éléments binaires définie, pour tout $n \geq 0$, par :

$$\begin{cases} u_0 = 0 & \text{si } n = 0 \\ u_n = u_{\left(\frac{n}{2}\right)} & \text{si } n \text{ est pair} \\ u_n = 1 - u_{\left(\frac{n-1}{2}\right)} & \text{si } n \text{ est impair} \end{cases}$$

1. Définir la fonction **saisie()** permettant de saisir et retourner une valeur entière positive. Une exception de type « ValueError » est levée, dans le cas de la saisie d'une valeur de type non entier et le message "Valeur erronée ! resaisir un entier svp" sera affiché à l'utilisateur.
2. Définir la fonction **estPair(n)** qui teste la parité d'un entier n.
3. Définir la fonction **réursive terme_rec(n)** qui à partir du paramètre entier positif n retourne le terme de la suite u correspondant à l'indice n.

Exemple : terme_rec (3) retourne 0.

4. Définir la fonction **réursive listTermes(n)** qui retourne une liste contenant tous les termes de la suite u ayant un indice inférieur ou égal à n.

Exemple : listTermes (10) retourne la liste [0,1,1,0,1,0,0,1,1,0,0].

5. Définir la fonction **itérative estSous_suite(s,u)** qui retourne True si s est une sous suite de la liste u , et False sinon.

Exemple : estSous_suite([0,1,0,0,1,1] ,[0,1,1,0,1,0,0,1,1,0,0]) retourne True.

6. Définir la fonction **itérative posSous_suite(s,u)** qui retourne un tuple formé par les positions de début et de fin de l'apparition de s dans u si s est une sous suite de u, sinon elle retourne un message " s n'est pas une sous suite de u ".

NB : si s appartient plusieurs fois dans u, les premières positions sont retournées.

Exemple : posSous_suite([0,1,0,0,1,1] ,[0,1,1,0,1,0,0,1,1,0,0]), la fonction retourne (3,8).

Exercice 2 : (12 points)

Dans le cadre d'un programme de santé et de bien-être, une application informatique est nécessaire pour aider les utilisateurs à évaluer leur poids et à suivre leur progression vers un poids idéal pour une bonne santé. Il s'agit, dans ce cas, de concevoir un système informatique qui permettra aux utilisateurs de :

- Entrer leurs données personnelles telles que le sexe, le poids et la taille.
- Enregistrer leur poids actuel et suivre les fluctuations de poids au fil du temps.
- Proposer des recommandations personnalisées pour atteindre un poids idéal en fonction des objectifs de santé individuels.

1. Définir la fonction **sauvegarde(n)** qui permet de sauvegarder n mesures, saisies au clavier, d'une personne dans un fichier texte nommé '**mesures.txt**'.

- ✓ La première ligne du fichier contient le sexe de la personne saisi comme une chaîne de caractères ayant la valeur 'F' (Féminin) ou 'M' (Masculin).
- ✓ Pour les autres (n-1) lignes, chaque ligne contient, dans l'ordre, les trois données suivantes séparées par le caractère ';' :
- **Annee** : un entier qui représente une année entre 1900 et 2024.
- **Taille** : un réel qui représente la taille normale en cm entre 120 et 200.
- **Poids** : un réel qui représente le poids en kg entre 25 et 150.

Exemple : voici un exemple du fichier '**mesures.txt**' contenant les lignes suivantes :

```
M
2015 ;159 ;42.5
2016 ;160 ;45.0
2017 ;162 ;50.7
2018 ;168 ;61.0
```

2. Définir la fonction **poidsParfait(T,S)** qui retourne le poids parfait d'une personne (en kg) selon sa taille T (en cm) et son sexe S, calculé par les deux formules suivantes:

- **Poids parfait Masculin** = $Taille - 100 - ((Taille - 150) / 4)$
- **Poids parfait féminin** = $Taille - 100 - ((Taille - 150) / 2.5)$

3. Définir la fonction **analyseDonnees()** qui, à partir du fichier '**mesures.txt**', lit les mesures pour retourner une **liste de tuples**. Chaque tuple contient deux éléments :

- ✓ Le premier élément est le résultat de comparaison entre le poids réel et le poids parfait de la personne noté par :
 - '**Inf**' : si le poids en kg de la personne est inférieur à son poids parfait
 - '**Sup**' : si le poids en kg de la personne est supérieur à son poids parfait
 - '**Egal**' : si le poids en kg de la personne est égal à son poids parfait
- ✓ Le deuxième élément est la différence entre les deux poids.

Une exception de type « **FileNotFoundException** » est levée, dans le cas où le fichier '**mesures.txt**' est inexistant et le message "FICHIER INTROUVABLE !" sera affiché à l'utilisateur.

Pour l'exemple donné dans la question 1, l'appel de la fonction **analyseDonnees()** retourne **[(' Inf',14.25), (' Inf',12.5), (' Inf',8.3), (' Inf',2.5)]**