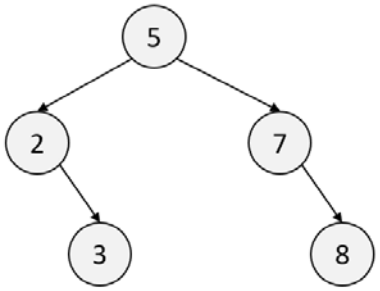
EXERCICE 1 : Arbre binaire de recherche (4 points)

Cet exercice traite principalement du thème « algorithmique, langages et programmation » et en particulier les arbres binaires de recherche. La première partie aborde les arbres en mode débranché via l'application d'un algorithme sur un exemple. La suivante porte sur la programmation orientée objet. La dernière partie fait le lien avec les algorithmes de tri.

Partie A : Étude d'un exemple

Considérons l'arbre binaire de recherche ci-dessous.



1. Indiquer quelle valeur a le nœud racine et quels sont les fils de ce nœud.
2. Indiquer quels sont les nœuds de la branche qui se termine par la feuille qui a pour valeur 3.
3. Dessiner l'arbre obtenu après l'ajout de la valeur 6.

Partie B : Implémentation en Python

Voici un extrait d'une implémentation en Python d'une classe modélisant un arbre binaire de recherche.

```
class ABR:
    """Implémentation d'un arbre binaire de recherche (ABR)"""
    def __init__(self, valeur=None):
        self.valeur = valeur
        self.fg = None
        self.fd = None

    def estVide(self):
        return self.valeur == None

    def insererElement(self, e):
        if self.estVide():
            self.valeur = e
        else:
            if e < self.valeur:
                if self.fg:
```

```
                    self.fg.insererElement(e)
                else:
                    self.fg = ABR(e)
            if e > self.valeur:
                if self.fd:
                    self.fd.insererElement(e)
                else:
                    self.fd = ABR(e)
```

1. Expliquer le rôle de la fonction `__init__`.

2. Dans cette implémentation, expliquer ce qui se passe si on ajoute un élément déjà présent dans l'arbre.

3. Recopier et compléter les lignes de code surlignées ci-dessous permettant de créer l'arbre de la partie A.

```
arbre = ABR(.....)
arbre.insererElement(2)
arbre.insererElement(.....)
arbre.insererElement(7)
arbre.insererElement(.....)
```

Partie C : Tri par arbre binaire de recherche

On souhaite trier un ensemble de valeurs entières distinctes grâce à un arbre binaire de recherche. Pour cela, on ajoute un à un les éléments de l'ensemble dans un arbre initialement vide. Il ne reste plus qu'à parcourir l'arbre afin de lire et de stocker dans un tableau résultat les valeurs dans l'ordre croissant.

1. Donner le nom du parcours qui permet de visiter les valeurs d'un arbre binaire de recherche dans l'ordre croissant.

2. Comparer la complexité de cette méthode de tri avec celle du tri par insertion ou du tri par sélection.

EXERCICE 2 (4 points)

Principaux thèmes abordés : bases de données (modèle relationnel, base de données relationnelle et langage SQL).

Dans notre monde, l'information a de plus en plus de valeur et d'importance mais nous sommes de plus en plus confrontés à l'infobésité.

Considérons l'utilisation des données issues de la table de Mendeleïev (tableau périodique des éléments). Il est contraignant de faire des recherches sur des moteurs dédiés à chaque fois qu'une valeur est nécessaire (masse volumique, rayon de covalence, point de fusion…).

Les lignes 3, 4 et 5 de cette table Mendeleïev ont permis de construire, en **annexe 1 de l'exercice 2,** une base de données des différents atomes correspondants.

1. Donner le nom du langage informatique utilisé pour accéder aux données dans une base de données ?

2. a) Lister les différents attributs des tables ATOMES et VALENCE en précisant le type du domaine de chacun.

b) Déterminer si des attributs de la table ATOMES peuvent avoir un rôle de clé primaire et/ou de clé étrangère. Justifier.

c) Donner le schéma relationnel pour les deux tables ATOMES et VALENCE.

3. Donner les réponses des deux requêtes suivantes :

a) `SELECT nom FROM ATOMES WHERE L='3' ORDER BY Sym`

b) `SELECT DISTINCT C FROM ATOMES`

4. Donner la requête SQL:

a) Pour afficher le nom et la masse atomique des atomes.

b) Pour afficher le symbole des atomes dont la couche de valence est s.

5. On a remarqué une erreur de saisie dans la table ATOMES, la masse atomique de l'argon (Ar) n'est pas 29,948 $g.mol^{-1}$ mais 39,948 $g.mol^{-1}$. Écrire la requête SQL pour corriger cette erreur de saisie.

Exercice 2 - Annexe 1

Relation « ATOMES »

Z	nom	Sym	L	C	masse_atom
11	sodium	Na	3	1	22,9897693
12	magnesium	Mg	3	2	24,305
13	aluminium	Al	3	13	26,9815386
14	silicium	Si	3	14	28,0855
15	phosphore	P	3	15	30,973762
16	soufre	S	3	16	32,065
17	chlore	Cl	3	17	35,453
18	argon	Ar	3	18	**29,948**
19	potassium	K	4	1	39,0983
20	calcium	Ca	4	2	40,078
21	scandium	Sc	4	3	44,955912
22	titane	Ti	4	4	47,867
23	vanadium	V	4	5	50,9415
24	chrome	Cr	4	6	51,9961
25	manganese	Mn	4	7	54,938045
26	fer	Fe	4	8	55,845
27	cobalt	Co	4	9	58,933195
28	nickel	Ni	4	10	58,6934
29	cuivre	Cu	4	11	63,546
30	zinc	Zn	4	12	65,409
31	gallium	Ga	4	13	69,723
32	germanium	Ge	4	14	72,64
33	arsenic	As	4	15	74,9216
34	selenium	Se	4	16	78,96
35	brome	Br	4	17	79,904
36	krypton	Kr	4	18	83,798
37	rubidium	Rb	5	1	85,4678
38	strontium	Sr	5	2	87,62
39	yttrium	Y	5	3	88,90585
40	zirconium	Zr	5	4	91,224
41	niobium	Nb	5	5	92,90638
42	molybdene	Mo	5	6	95,94
43	technetium	Tc	5	7	98
44	ruthenium	Ru	5	8	101,07
45	rhodium	Rh	5	9	102,9055
46	palladium	Pd	5	10	106,42
47	argent	Ag	5	11	107,8682
48	cadmium	Cd	5	12	112,411
49	indium	In	5	13	114,818
50	etain	Sn	5	14	118,71
51	Antimoine	Sb	5	15	121,76
52	Tellure	Te	5	16	127,6
53	Iode	I	5	17	126,90447
54	Xenon	Xe	5	18	131,293

Relation « VALENCE »

Col	Couche
1	s
2	s
3	d
4	d
5	d
6	d
7	d
8	d
9	d
10	d
11	d
12	d
13	p
14	p
15	p
16	p
17	p
18	p

Z : Numéro atomique ;
Sym : symbole ;
L : lignes ;
C ou Col : Colonne ;
Couche : Couche de valence

EXERCICE 3 (4 points)

Principaux thèmes abordés : structure de données (programmation objet) et langages et programmation (spécification).

Une entreprise fabrique des yaourts qui peuvent être soit nature (sans arôme), soit aromatisés (fraise, abricot ou vanille).

Pour pouvoir traiter informatiquement les spécificités de ce produit, on va donc créer une classe *Yaourt* qui possèdera un certain nombre d'attributs :

- Son genre : nature ou aromatisé
- Son arôme : fraise, abricot, vanille ou aucun
- Sa date de durabilité minimale (DDM) exprimée par un entier compris entre 1 et 365 (on ne gère pas les années bissextiles). Par exemple, si la DDM est égale à 15, la date de durabilité minimale est le 15 janvier.

On va créer également des méthodes permettant d'interagir avec l'objet `Yaourt` pour attribuer un arôme ou récupérer un genre par exemple. On peut représenter cette classe par le tableau de spécifications ci-dessous :

Yaourt	
ATTRIBUTS: - genre - arome - duree	**METHODES**: - Construire(arome,duree) - Obtenir_Arome() - Obtenir_Genre() - Obtenir_Duree() - Attribuer_Arome(arome) - Attribuer_Duree(duree) - Attribuer_Genre(arome)

1. La classe `Yaourt` est déclarée en Python à l'aide du mot-clé class :

```
class Yaourt:
    """ Classe définissant un yaourt caractérisé par :
    - son arome
    - son genre
    - sa durée de durabilité minimale"""
```

L'annexe 1 de l'exercice 3 donne le code existant et l'endroit des codes à produire dans les questions suivantes.

a) Quelles sont les assertions à prévoir pour vérifier que l'arôme et la durée correspondent bien à des valeurs acceptables. Il faudra aussi expliciter les commentaires qui seront renvoyés.

Pour rappel :

- L'arôme doit prendre comme valeur 'fraise', 'abricot', 'vanille' ou 'aucun'.
- Sa date de durabilité minimale (DDM) est une valeur positive.

b) Pour créer un yaourt, on exécutera la commande suivante :

```
Mon_Yaourt = Yaourt('fraise',24)
```

Quelle valeur sera affectée à l'attribut genre associé à `Mon_Yaourt` ?

c) Écrire en python une fonction `GetArome(self)`, renvoyant l'arôme du yaourt créé.

2. On appelle mutateur une méthode permettant de modifier un ou plusieurs attributs d'un objet.

Sur votre copie, écrire en Python le mutateur `SetArome(self,arome)` permettant de modifier l'arôme du yaourt.

On veillera à garder une cohérence entre l'arôme et le genre.

3. On veut créer une pile contenant le stock de yaourts. Pour cela il faut tout d'abord créer une pile vide :

```
def creer_pile():
    pile = [ ]
    return pile
```

a) Créer une fonction `empiler(p,Yaourt)` qui renvoie la pile p après avoir ajouté un objet de type `Yaourt` à la fin.

b) Créer une fonction `depiler(p)` qui renvoie l'objet à dépiler.

c) Créer une fonction `estVide(p)` qui renvoie `True` si la pile est vide et `False` sinon.

d) Qu'affiche le bloc de commandes suivantes ci-dessous ?

```
mon_Yaourt1 = Yaourt('aucun',18)
mon_Yaourt2 = Yaourt('fraise',24)
ma_pile = creer_pile()
empiler(ma_pile, mon_Yaourt1)
empiler(ma_pile, mon_Yaourt 2)
print(depiler(ma_pile).GetDuree())
print(estVide(ma_pile))
```

Exercice 3 - Annexe 1

```
class Yaourt:

    def __init__(self,arome,duree):
    # **** Assertions : question 1.a. à compléter sur votre copie
        self.__arome = arome
        self.__duree = duree
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'


    # **** Méthode GetArome(self) question 1.c. à compléter sur
    votre copie

    def GetDuree(self):
        return(self.__duree)

    def GetGenre(self):
        return ( self.__genre)

    def SetDuree(self,duree):
    # **** Mutateur de durée
        if duree > 0 :
            self.__duree = duree

    # **** Mutateur d'arôme SetArome(self,arome) - question 2. à
compléter sur votre copie

    def __SetGenre(self,arome):
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'
```