L'usage de la calculatrice n'est pas autorisé

Exercice 1

arbres binaires de recherche, langage objet

Dans cet exercice, les arbres binaires de recherche **ne peuvent pas comporter plusieurs fois la même clé**. De plus, **un arbre binaire de recherche limité à un nœud a une hauteur de 1**. On considère l'arbre binaire de recherche représenté ci-dessous, où `val` représente un entier :

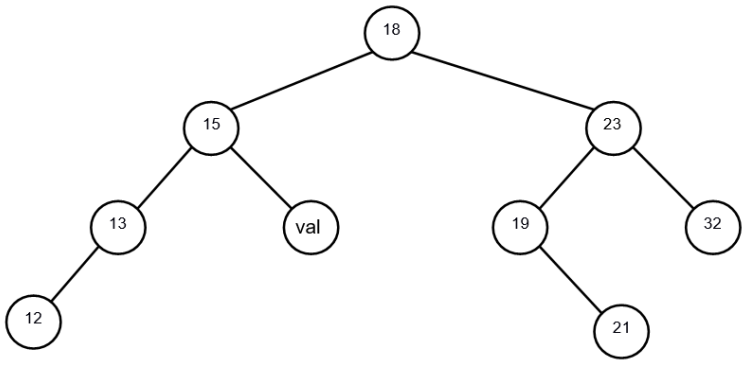

Figure 1

1. **a.** Donner le nombre de feuilles de cet arbre et préciser leur valeur (étiquette).

   b. Représenter le sous arbre-gauche du nœud 23.

   c. Donner la hauteur et la taille de l'arbre.

   d. Donner les valeurs entières possibles de `val` pour cet arbre binaire de recherche.

2. On suppose, pour la suite de cet exercice, que `val` est égal à 16.

   On rappelle qu'un parcours infixe depuis un nœud consiste, dans l'ordre, à faire un parcours infixe sur le sous arbre-gauche, afficher le nœud puis faire un parcours infixe sur le sous-arbre droit. Dans le cas d'un parcours suffixe (aussi appelé postfixe), on fait un parcours suffixe sur le sous-arbre gauche puis un parcours suffixe sur le sous-arbre droit, avant d'afficher le nœud.

   a. Donner les valeurs d'affichage des nœuds dans le cas du parcours infixe de l'arbre.

   b. Donner les valeurs d'affichage des nœuds dans le cas du parcours suffixe de l'arbre.

3. On considère la classe **Noeud** définie ainsi :

Code Python

```
class Noeud():
    def __init__(self, v):
        self.ag = None
        self.ad = None
        self.v = v

    def insere(self, v):
        n = self
        est_insere = False
        while not est_insere :
            if v == n.v:
                est_insere = True       # Bloc 1 (une ligne)
            elif v < n.v:
                if n.ag != None:        # Bloc 2
                    n = n.ag            #
                else:                   #
                    n.ag = Noeud(v)     #
                    est_insere = True   # Fin bloc 2
            else:
                if n.ad != None:        # Bloc 3
                    n = n.ad            #
                else:                   #
                    n.ad = Noeud(v)     #
                    est_insere = True   # Fin bloc 3

    def insere_tout(self, vals):
        for v in vals:
            self.insere(v)
```

a. Représenter l'arbre construit suite à l'exécution des instructions suivantes :

Code Python

```
racine = Noeud(18)
racine.insere_tout([12, 13, 15, 16, 19, 21, 32, 23])
```

b. Écrire les deux instructions permettant de construire l'arbre de la figure 1. On rappelle que le nombre `val` est égal à 16.

c. On considère l'arbre tel qu'il est présenté sur la figure 1.
Déterminer l'ordre d'exécution des blocs (repérés de 1 à 3) suite à l'application de la méthode `insere(19)` au nœud `racine` de cet arbre.

d. Écrire une méthode `recherche(self, v)` qui prend en argument un entier `v` et renvoie la valeur `True` si cet entier est une étiquette de l'arbre, `False` sinon.

Exercice 2

gestion des processus

Partie A

Cette partie est un questionnaire à choix multiples (QCM).
Pour chacune des questions, une seule des quatre réponses est exacte. Le candidat indiquera sur sa copie le numéro de la question et la lettre correspondant à la réponse exacte. Aucune justification n'est demandée. Une réponse fausse ou une absence de réponse n'enlève aucun point.

1. Parmi les commandes ci-dessous, laquelle permet d'afficher les processus en cours d'exécution ?

   a. `dir` b. `ps` c. `man` d. `ls`

2. Quelle abréviation désigne l'identifiant d'un processus dans un système d'exploitation de type UNIX ?

   a. `PIX` b. `SIG` c. `PID` d. `SID`

3. Comment s'appelle la gestion du partage du processeur entre différents processus ?

   a. interblocage b. ordonnancement c. planification d. priorisation

4. Quelle commande permet de supprimer un processus dans un système d'exploitation de type UNIX ?

   a. `stop` b. `interrupt` c. `end` d. `kill`

Partie B

1. Un processeur choisit à chaque cycle d'exécution le processus qui doit être exécuté. Le tableau ci-dessous donne pour trois processus P1, P2, P3 :

   – la durée d'exécution (en nombre de cycles);

– l'instant d'arrivée sur le processeur (exprimé en nombre de cycles à partir de 0);

– le numéro de priorité.

Le numéro de priorité est d'autant plus petit que la priorité est grande.
On suppose qu'à chaque instant, c'est le processus qui a le plus petit numéro de priorité qui est exécuté, ce qui peut provoquer la suspension d'un autre processus, lequel reprendra lorsqu'il sera le plus prioritaire.

Processus	Durée d'exécution	Instant d'arrivée	Numéro de priorité
P1	3	3	1
P2	3	2	2
P3	4	0	3

Reproduire le tableau ci-dessous sur la copie et indiquer dans chacune des cases le processus exécuté à chaque cycle.

2. On suppose maintenant que les trois processus précédents s'exécutent et utilisent une ou plusieurs ressources parmi R1, R2 et R3.
Parmi les scénarios suivants, lequel provoque un interblocage ? Justifier.

Scénario 1	Scénario 2	Scénario 3
P1 acquiert R1	P1 acquiert R1	P1 acquiert R1
P2 acquiert R2	P2 acquiert R3	P2 acquiert R2
P3 attend R1	P3 acquiert R2	P3 attend R2
P2 libère R2	P1 attend R2	P1 attend R2
P2 attend R1	P2 libère R3	P2 libère R2
P1 libère R1	P3 attend R1	P3 acquiert R2

Exercice 3

binaire et logique

Pour chiffrer un message, une méthode, dite du masque jetable, consiste à le combiner avec une chaîne de caractères de longueur comparable. Une implémentation possible utilise l'opérateur XOR (ou exclusif) dont voici la table de vérité :

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

Dans la suite, les nombres écrits en binaire seront précédés du préfixe `0b`.

1. Pour chiffrer un message, on convertit chacun de ses caractères en binaire (à l'aide du format UNICODE), et on réalise l'opération XOR bit à bit avec la clé.

   Après conversion en binaire, et avant que l'opération XOR bit à bit avec la clé n'ait été effectuée, Alice obtient le message suivant :

   `m = 0b 0110 0011 0100 0110`

   a. Le message `m` correspond à deux caractères codés chacun sur 8 bits : déterminer quels sont ces caractères. On fournit pour cela la table ci-dessous qui associe à l'écriture hexadécimale d'un octet le caractère correspondant.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	space	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

   Exemple de lecture : le caractère correspondant à l'octet codé 4A *en hexadécimal est la lettre* J.

   b. Pour chiffrer le message d'Alice, on réalise l'opération XOR bit à bit avec la clé suivante :

   `k = 0b 1110 1110 1111 0000`

   Donner l'écriture binaire du message obtenu.

2. a. Donner la table de vérité de l'expression booléenne `(a XOR b) XOR b`.

   b. Bob connaît la chaîne de caractères utilisée par Alice pour chiffrer le message. Quelle opération doit-il réaliser pour déchiffrer son message ?