

Nom	
Prénom	
Groupe	

Note	18/20
------	-------

Algorithmique
Arbres binaires et généraux
 SUP S2 EPITA
Examen B3
 12 mars 2024

7/7
 5/7
 3/3
 3/3

Consignes (à lire) :

- ☐ Vous devez répondre directement **sur ce sujet**.
 - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées**.
 - Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Code :**
 - Tout code doit être écrit dans le langage Python (pas de C, CAML, ALGO ou autre).
 - **Tout code Python non indenté ne sera pas corrigé.**
 - Tout ce dont vous avez besoin (types, fonctions, méthodes) est indiqué en **annexe**.
 - Vos fonctions doivent impérativement respecter les exemples d'applications donnés.
 - Vous pouvez également écrire vos propres fonctions, dans ce cas elles doivent être documentées (on doit savoir ce qu'elles font).
 - Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.
 - Comme d'habitude l'optimisation est notée. Si vous écrivez des fonctions non optimisées, vous serez notés sur moins de points.¹
- ☐ Durée : 1h30

Exercice 1 (Croissant— 7 points)

Écrire la fonction `increasing(B)` qui vérifie que chaque niveau de l'arbre binaire `B` contient strictement plus de nœuds que le niveau précédent.

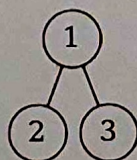


FIGURE 1 – B4

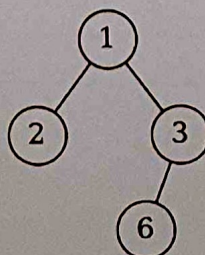


FIGURE 2 – B5

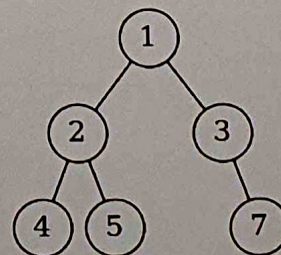


FIGURE 3 – B6

Exemples d'applications :

```

1  >>> increasing(None)
2  True
3  >>> increasing(B4)
4  True
5  >>> increasing(B5)
6  False
7  >>> increasing(B6)
8  True
  
```

1. Des fois, il vaut mieux moins de points que pas de points.

```
def increasing(B):  
    if not B:  
        return True  
    q = Queue()  
    p = Queue()  
    q.enqueue(B)  
    cpt, max, endLevel = 0, -1, True  
    while not q.isEmpty() and endLevel:  
        cur = q.dequeue()  
        cpt += 1  
        if B.left:  
            p.enqueue(B.left)  
        if B.right:  
            p.enqueue(B.right)  
        if q.isEmpty():  
            p.q = q, p  
            if cpt > max:  
                cpt, max = 0, cpt  
            else:  
                endLevel = False  
    return endLevel
```

Exercice 2 (Somme - 7 points)

Écrire la fonction `check_sum(B)` qui vérifie si chaque point double de l'arbre binaire `B` contient la somme des clés de ses deux fils. Le **parcours profondeur** doit obligatoirement être utilisé.



FIGURE 4 - B1

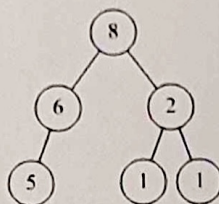


FIGURE 5 - B2

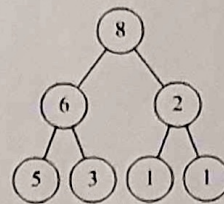


FIGURE 6 - B3

Exemples d'applications :

```

1  >>> check_sum(None)
2  True
3  >>> check_sum(B1)
4  True
5  >>> check_sum(B2)
6  True
7  >>> check_sum(B3)
8  False
  
```

gph: gestion des None rest -2

```

def check_sum(B):
    if not B:
        return True
    if not B.left and not B.right:
        return True
    if B.left and B.right:
        return B.key == B.left.key + B.right.key and check_sum(B.left)
        and check_sum(B.right)
    if B.left:
        return check_sum(B.left)
    return check_sum(B.right)
  
```

Exercice 3 (Mystery– 3 points)

```

1 def mystery(B):
2     if B == None:
3         return (None, 0)
4     else:
5         C = BinTree(0, None, None)
6         C.left, b = mystery(B.left)
7         if B.left != None:
8             if B.right != None:
9                 b = b + 1
10            C.right, d = mystery(B.right)
11            C.key = b + d
12            return (C, C.key)

```

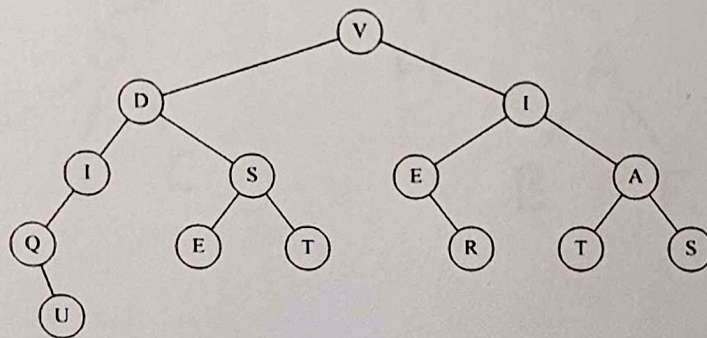
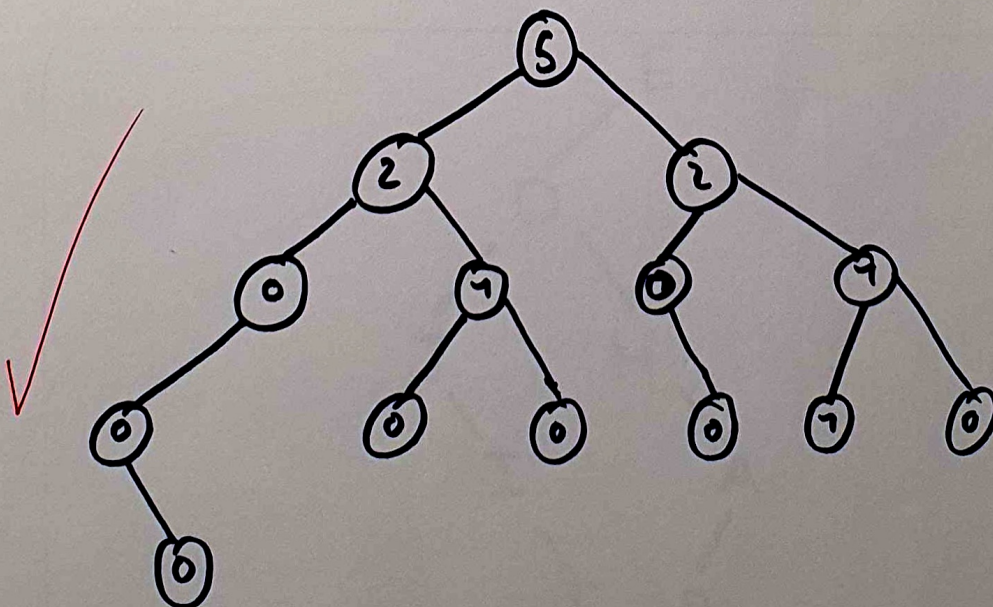


FIGURE 7 – Bquid

Dessiner ci-dessous l'arbre résultat de l'application de `mystery(Bquid)`.



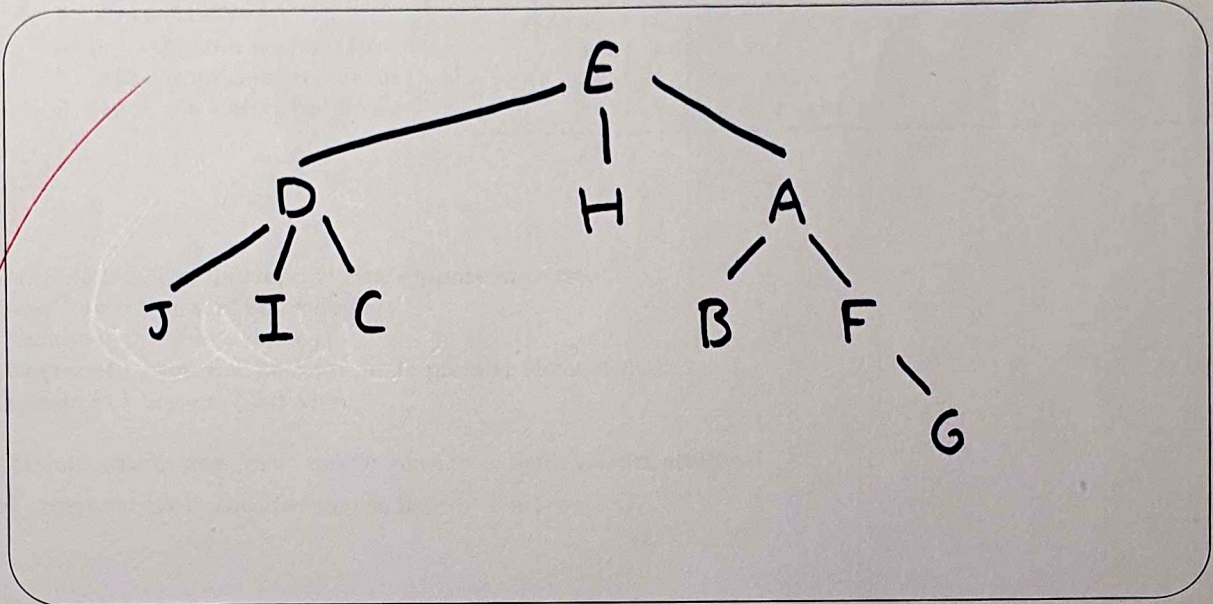
Exercice 4 (Arbre général- 3 points)

Soit un arbre général A. Les traitements préfixes et suffixes du parcours profondeur de A affichent les séquences suivantes :

préfixe = E D J I C H A B F G

suffixe = J I C D H B G F A E

— Dessiner ci-dessous l'arbre général A.



— Dessiner ci-dessous l'arbre général A sous forme binaire premier fils - frère droit.

