165

Nom	
Prénom	
Groupe	B2

Note	19,5

Algorithmique
INFO-SPÉ - S3
Contrôle n° 3 (C3)
23 octobre 2023
Feuilles de réponses

1	6
2	5
3	5,5
4	3

Réponses 1 **(Haches et graphes... – *6 points*)**

1. Donnez une méthode de hachage indirect :

   hachage coalescent 0,5

2. Le problème qui apparait avec le hachage coalescent est :

   Cela peut provequer des collisions secondaires 0,5

3. Une collision primaire est :

   une collision entre deux elements x et y distincts avec $v = h(x) = h(y)$ (même valeur de hachage)

   0,5

4. L'ordre d'un graphe non orienté est :

   Son nombre de sommets 0,5

   1

5. Un sommet isolé est :

   Un sommet de degré 0 0,5

6. Le tableau des demi-degrés extérieurs des sommets de G :

	1	2	3	4	5	6	7	8	9
demi-degrés extérieurs	2	2	3	2	0	2	3	2	2

   1

7. Le graphe G est-il fortement connexe ? OUI (NON) 0,5

8. Si NON, combien possède t-il de composantes fortement connexes ? 2 1

9. S'ils existent, les sommets de G de degré différent de 4 sont : 1, 5, 9 0,5

10. S'ils existent, les sommets de G de demi-degré intérieur égal à 0 sont : / 0,5

Réponses 2 **(Level from x – *5 points*)**

Spécifications :

`keys_after(T:Tree, x:int)` retourne la liste des clés dans l'arbre `T` après la **première** valeur `x` trouvée sur le même niveau que `x`. Elle retourne une liste vide si `x` n'est pas dans `T` ou s'il n'y a pas de nœud après.

```
def keys_after(T:Tree, x:int):
    q = Queue()
    q.enqueue(T)
    q.enqueue(None)
    L = []
    while not q.isempty():
        N = q.dequeue()
        if N != None:
            if N.Key == x :
                N = q.dequeue()
                while N != None:
                    L.append(N.Key)
                    N = q.dequeue()
                Return L
            for child in N.children:
                q.enqueue(child)
        else:
            if not q.isempty():
                q.enqueue(None)
            else:
                return L
```

5/5

Réponses 3 **(Average subtrees –** ***6 points*****)**

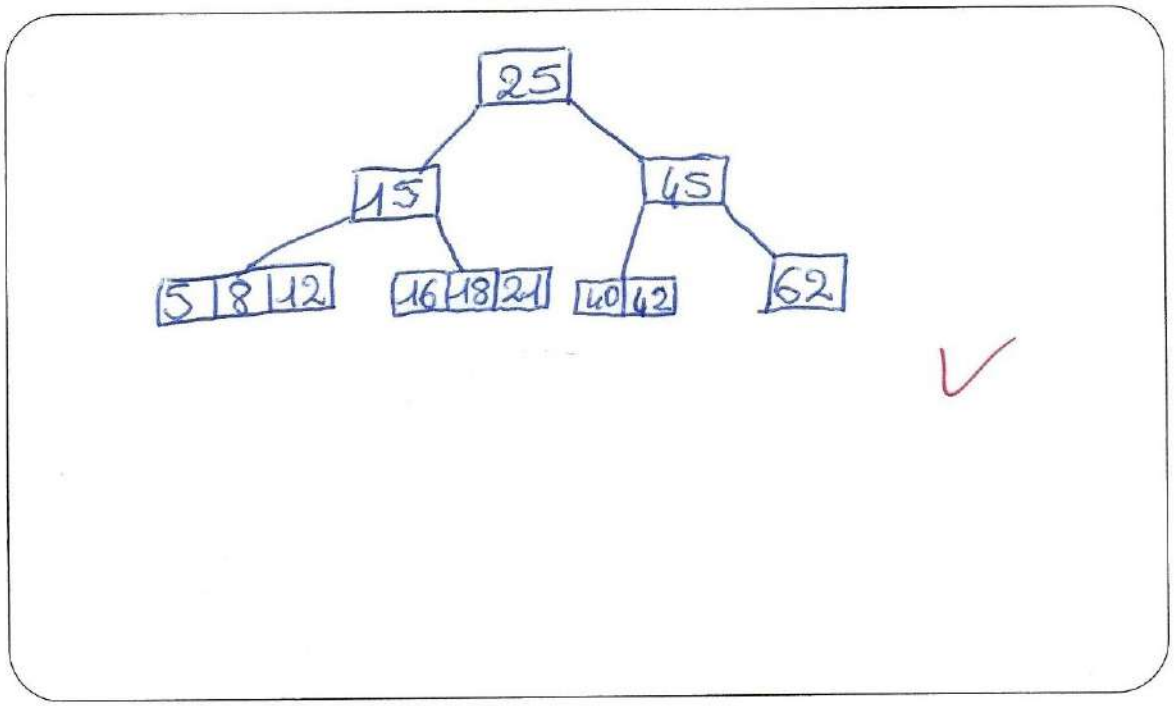
Spécifications :

`average_subtrees(B:TreeAsBin)` vérifie si aucun nœud de l'arbre `B` n'a un sous-arbre dont la moyenne des clés est strictement supérieure à la clé de ce nœud.

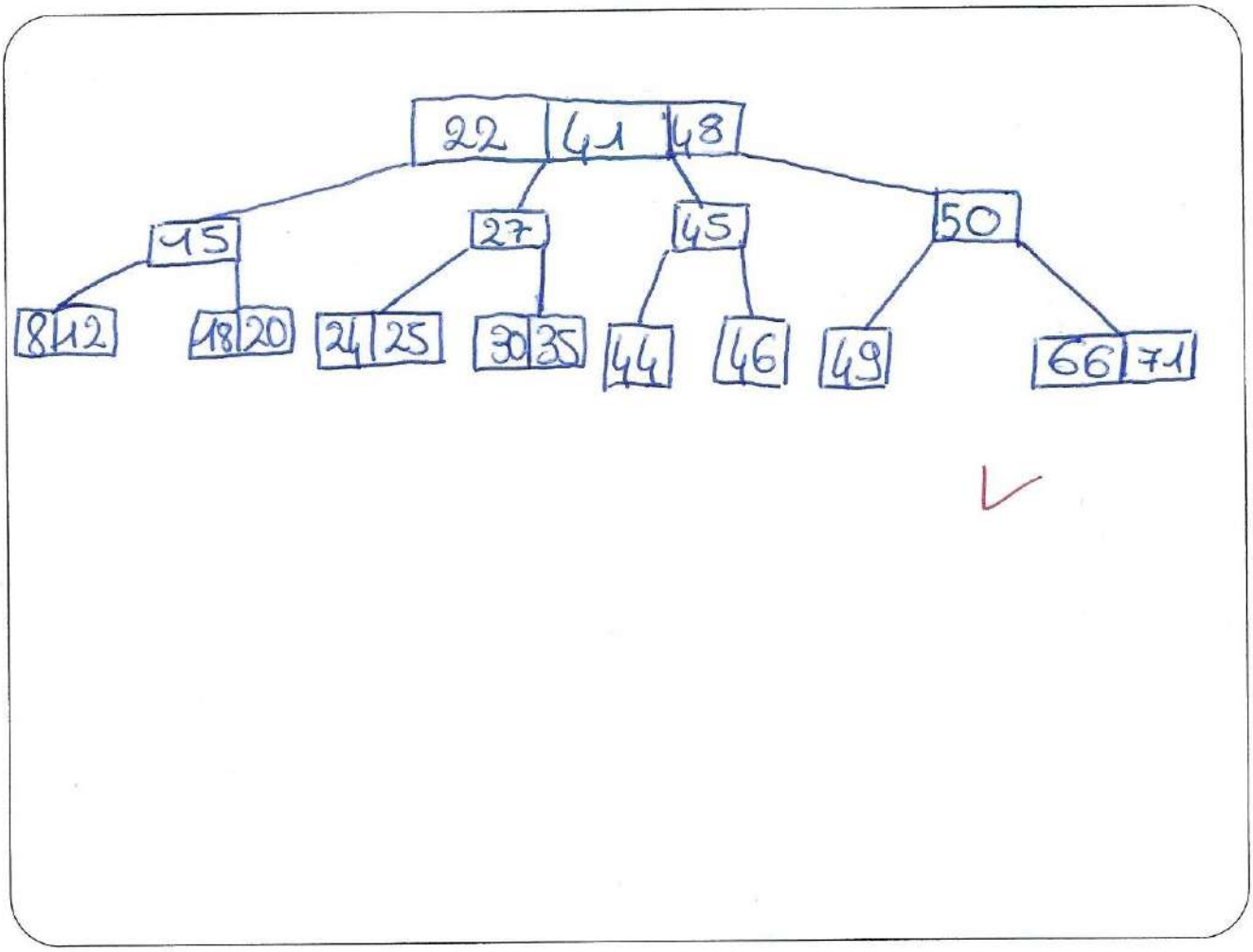
```
"""Verifie si aucun noeud de B(TreeAsBin) n'a un sous-arbre
dont la moyenne des clés est strictement supérieure à
la clé de ce noeud.
Si ce n'est pas verifié alors la fonction
retourne (-1,-1) sinon elle retourne la somme
des clés d'un sous-arbre et le nombre de noeuds
dans celui-ci. Paramètre : B (TreeAsBin)
return (int,int)
"""

def __aux_average_sub(B:TreeAsBin):
    """
    C=B.child
    (nb,K)=(0, 0)
    while C!=None:
        (Ka, nb2)= __aux_average_sub(C)
        if nb2==-1 or Ka/nb2 > B.key:
            return (-1,-1)
        K+=Ka
        nb+=nb2
        C=C.sibling
    return (K+B.key, nb+1)

def average_subtrees(B:TreeAsBin):
    if B!=None:
        (Ka,nb2)= __aux_average_sub(B)
        if nb2 == -1:
            return False
    return True
```

Réponses 4 **(B-arbre : insertions et suppression –** *3 points*)

1. L'arbre B1 après insertions des valeurs 21, 42 et 8 :



2. Arbre B2 après suppression de la valeur 80 :



3/3