Proposition de correction

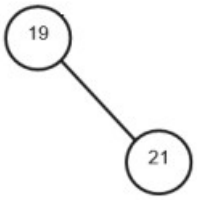
Exercice 1

Q1.a

4 feuilles :

- 12
- val
- 21
- 32

Q1.b



Q1.c

hauteur : 4

taille : 9

Q1.d

Val : [16, 17]

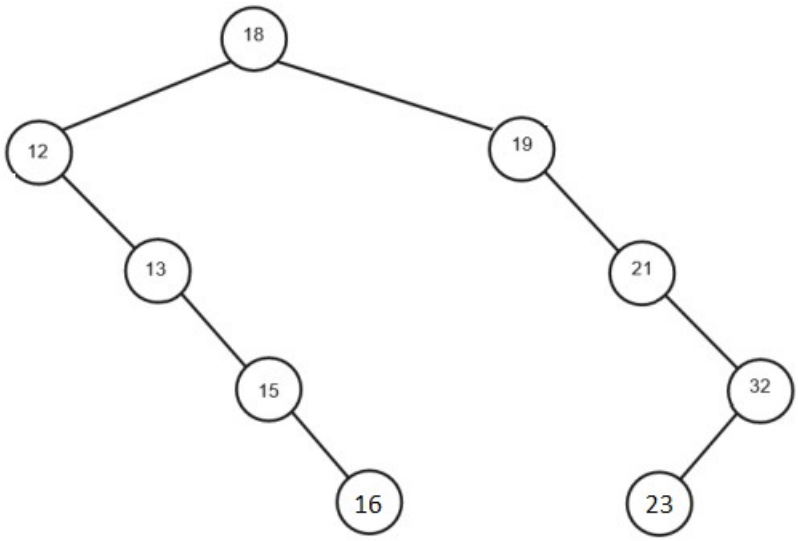
Q2.a

12, 13, 15, 16, 18, 19, 21, 23, 32 (valeurs triées)

Q2.b

12, 13, 16, 15, 21, 19, 32, 23, 18

Q3.a



Q3.b

racine = Noeud(18)

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

Q3.c

bloc 3 – bloc 2

ne rentre pas dans bloc 1

Q4

```
def recherche(self, v):
    n = self
    while n is not None :
        if v == n.v:
            return True
        elif v < n.v:
            n = n.ag
        else:
            n = n.ad
    return False
```

Exercice 2

Partie A

Q1

ps

Q2

PID

Q3

L'ordonnancement

Q4

kill

Partie B

Q1

P3	P3	P2	P1	P1	P1	P2	P2	P3	P3

Q2

Scénario 2

P1 ← R1
P3 ← R2
P1 ← R2
P3 ← R1 R1 jamais libéré par P1

Partie C

Q1.a

m = 0b 0110 0011 0100 0110 = 0x 63 46 = cF

Q1.b

m = 0b 0110 0011 0100 0110

k = 0b 1110 1110 1111 0000

c = 0b 1000 1101 1011 0110

Q2.a

a	b	a XOR b	(a XOR b) XOR b
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Q2.b

Appliquer de nouveau l'opération XOR bit à bit avec la clé sur le message chiffré

Exercice 3

Q1

SGBDR : Système de Gestion des Bases de Données Relationnel

Q2.a

La première requête efface de la table Train le train n°1241. Or la deuxième requête fait référence (Reservation.numT=1241) à une entrée qui n'existe plus dans la base Train.

Q2.b

Si un ° de train n'existe pas, Il est impossible d'effectuer une réservation pour ce train.

Q3.a

```
SELECT numT
FROM Train
WHERE Destination = 'Lyon'
ORDER BY numT
```

Q3.b

```
INSERT INTO Reservation
VALUES(1307, 'Turing', 'Alan', 33, 7869)
```

Q3.c

```
UPDATE Train
SET horaireArrivee = '08:11'
WHERE numT = 7869
```

Q4

De dénombrer les réservations faites au nom de Grace Hopper

Q5

```
SELECT Train.destination, Reservation.prix
FROM Train, Reservation
WHERE Train.nomClient = 'Hopper' AND Train.prenomClient = 'Grace'
AND Train.numT = Reservation.numT
ORDER BY Train.destination
```

Exercice 4

Q1.a

O(n.ln2(n))

Q1.b

L'algorithme du tri à bulles a une complexité en temps en $O(n^2)$ dans le pire des cas et en O(n) si la liste est déjà trié.

$O(n^2) > O(n.ln2(n)) > O(n)$

Q2

[7, 4, 2, 1, 8, 5, 6, 3]

[7, 4, 2, 1]

[7, 4]

[2, 1]

[8, 5, 6, 3]

[8, 5]

[6, 3]

Q3

```
def moitie_droite(L : list) -> list:
    return L[len(L) // 2:]
```

ou

```
def moitie_droite(L : list) -> list:
    debut = len(L)//2
    tab   = []
    for i in range(debut, len(L)):
            tab.append(L[i])
    return tab
```

Q4

```
def fusion(L1, L2):
  L  = []
  n1 = len(L1)
  n2 = len(L2)
  i1 = 0
  i2 = 0
  while i1 < n1 or i2 < n2:
    if i1 >= n1:
      L.append(L2[i2])
```

```
            i2 = i2+1
        elif i2 >= n2:
            L.append(L1[i1])
            i1=i1+1
        else :
            e1 = L1[i1]
            e2 = L2[i2]
            if e1 > e2:
                L.append(e2)
                i2 = i2 + 1
            else :
                L.append(e1)
                i1 = i1 + 1
    return L
```

Exercice 5

Q1.a

86.154.10.1 : R2

Q1.b

R1 → R2 → R6

Q2.a

R1 → R3 → R4 → R6

Q2.b

Seule la ligne R1 sera modifiée : 86.154.10.0 (R2) → 112.44.65.0 (R3)

Q3.a

$$C = \frac{10^9}{BP} = \frac{10^9}{10 \times 10^6} = 100$$

Q3.b

R1 → R2 → R4 → R5 → R6

Q3.c

R2 et R4