Éléments de correction sujet 03

Exercice 1

1

```
  def __init__(self, nature, surface, prix_moy):
        self.nt = nature
        self.sf = surface
        self.pm = prix_moy
```

2

L'instruction b1.estim_prix() renvoie 140000.0 (estimation du prix de b1) ; la valeur renvoyée est de type flottant, car self.sf et self.pm sont de type flottant.

3

```
 def estim_prix(self):
        prix_brut = self.sf * self.pm
        if self.nt=='maison':
            return prix_brut*1.1
        elif self.nt=='bureau':
            return prix_brut*0.8
        else :
            return prix_brut
```

4

```
 def nb_maison(lst):
     compteur = 0
     for b in lst :
         if b.nt=='maison':
             compteur = compteur + 1
     return compteur
```

5a

on effectue un parcours infixe : b2 - b4 - b1 - b5 - b3 - b6

5b

```
 def contient(surface,abr):
     if abr.est_vide():
         return False
     elif abr.get_v().sf >= surface:
         return True
     else :
         return contient(surface, abr.get_d())
```

Exercice 2

1

R2 (utilisation de *)

2a

```
SELECT nom, avis
FROM Client
INNER JOIN Reservation ON  Client.idClient = Reservation.idClient
WHERE jour = '2021-06-05' AND heure = '19:30:00'
```

2b

```
SELECT nom
FROM Plat
INNER JOIN Commande ON Plat.idPlat = Commande.idPlat
INNER JOIN Reservation ON  Reservation.idReservation =
Commande.idReservation
WHERE (Categorie = 'plat principal' OR Categorie = 'dessert') AND jour =
'2021-04-12'
```

3

Cette requête permet d'ajouter un plat qui aura pour idPlat 58, pour nom "Pêche Melba", pour catégorie "dessert", pour description "Pêches et glace vanille" et pour prix 6,5 euros.

4a

```
DELETE FROM Commande
WHERE idReservation = 2047
```

4b

```
UPDATE Plat
SET prix = prix+0.05*prix
WHERE prix < 20.0
```

Exercice 3

1a

L1 : 192.168.1.0/24

L2 : 172.16.0.0/16

1b

réseau L1 : plus "petite : 192.168.1.1 ; plus "grande" : 192.168.1.254

réseau L2 : plus "petite" : 172.16.0.1 ; plus "grande" : 172.16.255.254

1c

réseau L1 : 256 - 2 = 254 adresses

réseau L2 : 256^2 - 2 = 2^{16} - 2 = 65534 adresses

2a

Il est utile d'avoir plusieurs chemins possibles en cas de panne (routeur ou connexion entre routeurs) ou encore en cas de trafic réseau trop important au niveau d'un routeur.

2b

Pour relier R1 à R6 il est possible d'effectuer seulement 2 sauts : R1 -> R2 -> R5 -> R6

2c

Liaison	R1-R2	R2-R5	R5-R6	R2-R3	R3-R4	R4-R5	R3-R5
coût	10	10	10	1	1	1	10

Le chemin reliant R1 à R6 ayant le plus petit coût est R1 -> R2 -> R3 -> R4 -> R5 -> R6

avec un coût de : 10 (R1-R2) + 1 (R2-R3) + 1 (R3-R4) + 1 (R4-R5) + 10 (R5-R6) = 23

3

R5 :

IP réseau de destination	Passerelle suivante	Interface
172.16.0.0/16	10.1.7.2	interface 4
192.168.1.0/24	10.1.3.1	interface 1

R6 :

IP réseau de destination	Passerelle suivante	Interface
10.1.7.1/24	10.1.7.1	interface 2
192.168.1.0/24	10.1.7.1	interface 2

Exercice 4

Partie A

1

Pour passer à l'exécution, le Programme 1 a besoin du modem, mais le modem est déjà utilisé par le Programme 2. Le processus p1 est donc bloqué. Pour passer à l'exécution, le Programme 2 a besoin de l'imprimante, mais l'imprimante est déjà utilisée par le Programme 3. Le processus p2 est donc bloqué. Pour passer à l'exécution, le Programme 3 a besoin de la table traçante,ElémentsElémentsEléments mais la table traçante est déjà utilisée par le Programme 1. Le processus p3 est donc bloqué. Les 3 processus sont bloqués et ne pourront pas libérer les ressources attendues par les autres processus (la libération des ressources se faisant après l'exécution), nous sommes bien en situation d'interblocage.

2

Programme 3
demander (table traçante)
demander (imprimante)
exécution
libérer (table traçante)
libérer (imprimante)

3

bloqué

Partie B

1

Elémentsps -ef

2

L'identifiant du processus parent à l'origine de tous les processus concernant le navigateur Web est 831

3

L'identifiant du processus dont le temps d'exécution est le plus long est 6211 (00:01:16)

Exercice 5

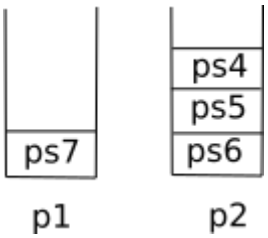
1

file

2

```
def ajouter(lst,proc):
    lst.append(proc)
```

3



4a

```
def est_vide(f):
    p1,  p2 = f
    return pile_vide(p1) and pile_vide(p2)
```

4b

```
def enfiler(f,elt):
    empiler(f[0],elt)
```

4c

```
def defiler(f):
    p1 = f[0]
    p2 = f[1]
    if pile_vide(p2):
        while not pile_vide(p1):
            v = depiler(p1)
            empiler(p2,v)
    return depiler(p2)
```