Éléments de correction
sujet 08

Exercice 1

1. La clé primaire doit être unique pour chaque entrée, or, on retrouve 2 fois le même numéro de série (id = 4 et id = 8). Donc num_ser ne peut pas être une clé primaire.
2.

Gibson	Les Paul Goldtop
Fender	Stratocaster

3.
```
SELECT annee
FROM inventaire
WHERE modele = 'Les Paul Standard'
```
4.
```
SELECT modele
FROM inventaire
WHERE marque = 'Gibson'
ORDER BY annee
```
5.
```
UPDATE inventaire
SET annee = 1957
WHERE id = 1
```
6. On doit d'abord créer la table marque car elle ne fait référence à aucune autre table. Puis, on doit créer la table modele, car elle fait référence à la table marque. La table table guitare doit être créée en dernier car elle fait référence à la table modele.
7.
```
SELECT num_ser, annee
FROM guitare
JOIN modele ON id_modele = guitare.id
WHERE nom = 'Les Paul Standard'
```
8.
```
DELETE
FROM guitare
WHERE id = 3
```
9.
```
INSERT INTO marque
(3,'BC Rich')

INSERT INTO modele
(5, 'Mockingbird', 3)

INSERT INTO guitare
(9, 5, 1992,'92R', 5000 )
```

10.
```
SELECT SUM(prix)
FROM guitare
JOIN modele id_modele = modele.id
WHERE nom = 'Stratocaster'
```

Exercice 2

1.
```
tache1 = Tache(1, 'Répondre aux e-mails', 45)
tache2 = Tache(2, 'Ranger ma chambre', 60)
```
2.
```
def avancer(self, n):
    self.duree_restante = self.duree_restante - n
```
3.
```
def est_terminee(self):
    return self.duree_restante == 0
```
4.
```
[début] (<t3>, 4) (<t7>, 4) (<t1>, 3) (<t2>, 3) (<t6>,2) (<t4>, 1)
(<t5>, 1) [fin]
```
5. valeur : <t3>
   file : [début](<t1>, 3)(<t2>, 3)(<t4>, 1)(<t5>, 1)[fin]
6.
   valeur : 4
   file : [début](<t3>, 4)(<t1>, 3)(<t2>, 3)(<t4>, 1)(<t5>, 1)[fin]
7.
```
def ajouter_file_prio(f, t, p):
    f_aux = File()
    while not f.est_vide() and f.examiner()[1] >= p:
        f_aux.enfiler(f.defiler())
    f_aux.enfiler((t,p))
    while not f.est_vide():
        f_aux.enfiler(f.defiler())
    while not f_aux.est_vide():
        f.enfiler(f_aux.defiler())
```
8. Le coût est O(n)
9.
   t3, t7, t3, t3, t3, t1, t2, t1, t2, t2, t6, t6, t6, t4, t5, t4, t5

10.
```
def planning(f):
    ordre = []
    while not f.est_vide():
        tache, prio = f.defiler()
        ordre.append(tache)
        tache.avancer(25)
        if not tache.est_terminee():
            ajouter_file_prio(f, tache, prio)
    return ordre
```

Exercice 3

1. (a) et (b)
2. 192.168.20.255
3. 256 - 2 - 4 = 250
4. $2^3 = 8$ il faut donc 3 bits pour avoir 8 adresses. On a 32 - 3 = 29 bits pour la partie réseau et 3 bits pour la partie machine.
5.

Routeur 2			
Réseau destination	interface sortie	prochain routeur	Nombre de sauts
…	…	…	…
192.168.30.0	172.16.4.1	172.16.4.2	1
172.16.1.0	172.16.3.1	172.16.3.2	1

6.

Routeur 2			
Réseau destination	interface sortie	prochain routeur	Nombre de sauts
…	…	…	…
192.168.10.0	172.16.4.1	172.16.4.2	2

7.

Routeur 2		
Réseau destination	interface sortie	prochain routeur
autre	172.16.3.1	172.16.3.2

8.

Tableau des coûts		
Type de connexion	Débit en bit/s	coût
Fast Ethernet	100 Mbit/s	10
Fibre optique	1 Gbit/s	1

9. 1 -> 2 -> 3 -> 4 ; coût = 10 + 10 + 1 = 21
10. ‘11000000.10101000.00010100.00001100’
11. Quand les adresses sont identiques
12.
```
def precede(ip_1, ip_2):
    for i in range(35):
        if ip_1[i] < ip_2[i]:
            return True
        elif ip_1[i] > ip_2[i]:
            return False
    return False
```
13. attribut : self.adresse_ip ; méthode : est_vide
14.
```
def est_vide(self):
    return self.adresse_ip == ''
```
15. Un arbre binaire de recherche permet d’avoir une recherche efficace, car au mieux, le coût est en O(log(n)).
16.
```
def modifie(self, adresse_ip,interface, passerelle,cout):
    if self.est_vide():
        self.gauche = Abr('','','',0)
        self.droite = Abr('','','',0)
    self.adresse_ip = adresse_ip
    self.interface = interface
    self.passerelle = passerelle
    self.cout = cout
```
17.
```
elif precede(ip_bin(adresse_ip), ip_bin(self.adresse_ip)):
```