Exercice 1 (6 points)

Correction

<table>
<tr><th>Question</th><th>Niveau</th><th>Contenu</th><th>Solution</th></tr>
<tr><td>1</td><td>1</td><td></td><td><code>num_serie</code> ne peut pas être une clé primaire de la relation <code>inventaire</code> car une clé primaire doit être unique. Or, deux guitares (de marques différentes) peuvent avoir le même numéro de série, par exemple la Gibson Les Paul Standard de 1987 et la Fender Stratocaster de 1965.</td></tr>
<tr><td>2</td><td>1</td><td></td><td>
<table>
<tr><th>marque</th><th>modele</th></tr>
<tr><td>Gibson</td><td>Les Paul Goldtop</td></tr>
<tr><td>Fender</td><td>Stratocaster</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>1</td><td></td><td>

```
SELECT annee
FROM inventaire
WHERE modele = 'Les Paul Standard'
```

</td></tr>
</table>

Question	*Niveau*	*Contenu*	*Solution*

4	2		`SELECT modele`
			`FROM inventaire`
			`WHERE marque = 'Gibson'`
			`ORDER BY annee`

5	2		`UPDATE inventaire`
			`SET annee = 1957`
			`WHERE id=1`

6	1		On commencera par créer la table `marque` puis la table `modele` et enfin la table `guitare`. En effet, l'attribut `id_marque` de la table `modele` faisant référence à l'attribut `id` de la table marque, il faut que celui-ci existe au préalable afin de respecter la contrainte de référence.

7	2		`SELECT g.num_ser, g.annee`
			`FROM guitare AS g`
			`JOIN modele AS mo ON g.id_modele = mo.id`
			`WHERE mo.nom = 'Les Paul Standard'`

Question	*Niveau*	*Contenu*	*Solution*
8	1		`DELETE FROM guitare AS g` `WHERE id = 3`

9	2		`INSERT INTO marque VALUES (3, 'BC RICH')`
			`INSERT INTO modele VALUES (5, 'Mockingbird', 3)`
			`INSERT INTO guitare VALUES (9, 5, 1988, '92R', 5000)`

10	3		`SELECT SUM(prix)`
			`FROM guitar AS g`
			`JOIN modele AS mo ON g.id_modele = mo.id`
			`WHERE mo.nom = 'Stratocaster'`

Exercice 2 (6 points)

Correction

Question	*Niveau*	*Contenu*	*Solution*
1	1	Programmation Orientée Objet	`tache1 = Tache(1, "Appeler maman.", 45)` `tache2 = Tache(2, "Ranger ma chambre.", 60)`
2	1	Écrire une méthode	`def avancer(self, n):` `    self.duree_restante -= n`
3	1	Écrire une méthode	`def est_terminee(self):` `    return self.duree_restante <= 0`
4	1	Modélisation	`[début] (<t3>, 4) (<t7>, 4) (<t1>, 3) (<t2>, 3) (<t6>, 2) (<t4>, 1) (<t5>, 1) [fin]`
5	1	File et tuple	`<t3>` `[début] (<t1>, 3) (<t2>, 3) (<t4>, 1) (<t5>, 1) [fin]`
6	1	File et tuple	`4` `[début] (<t3>, 4) (<t1>, 3) (<t2>, 3) (<t4>, 1) (<t5>, 1) [fin]`

Question	*Niveau*	*Contenu*	*Solution*

7	3	Manipulation de File	`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	2	Coût algorithmique	Le coût d'exécution temporel est linéaire, en $O(m)$.
9	2	Ordonnancement	t3, t7, t3, t3, t3, t1, t2, t1, t2, t2, t6, t6, t6, t4, t5, t4, t5

10	3	Traduire un algorithme en programme	`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 (8 points)

Correction

Question	*Niveau*	*Contenu*	*Solution*
1	1	Adressage IP	Les adresses (a) et (b).
2	1	Adressage IPV4	L'adresse de diffusion est 192.168.20.255.
3	1	Adressage IPV4	Sur un octet, il est possible d'obtenir 256 valeurs. Il existe donc 256 adresses machines au sein du réseau du café 1. En décomptant les deux adresses IP réservées (réseau et diffusion), les trois bornes de commandes et l'interface du routeur, il reste 250 adresses IP disponibles.
4	2	Adressage IPV4	Il faut exactement 3 bits pour coder 8 adresses différentes. Le masque pourrait donc aller jusqu'à compter 29 bits (32 bits - 3 bits).
5	1	Table de routage	Réseau Destination / Interface de sortie / Prochain routeur / nb sauts 192.168.30.0 / 172.16.4.1 / 176.16.4.2 / 1 172.16.1.0 / 172.16.3.1 / 172.16.3.2 / 1
6	1	Table de routage	On peut choisir, pour le réseau 192.168.10.0, la route qui sort par l'interface 172.16.4.1 vers le prochain routeur 172.16.4.2. On obtient encore une route à 2 sauts.

Question	*Niveau*	*Contenu*	*Solution*
7	1	Table de routage	Réseau destination / Interface de sortie / Prochain routeur autre / 172.16.3.1 / 172.16.3.2
8	1	Calcul du coût	Le coût de la liaison Fast Ethernet est 10 et celui de la fibre optique est 1.
9	1	Protocole OSPF	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ est la route à choisir pour respecter le protocole OSPF. Son coût est de 21.
10	1	Binaire	Les conversions en binaire des nombres 192 et 168 sont donnés dans l'exemple. Il reste à convertir 20 et 12 en binaire. '11000000.10101000.00010100.00001100'
11	2	Lire et interpréter un code	Il faut que tous les caractères soient identiques, c'est-à-dire que les adresses IP passées en paramètre soient les mêmes.
12	1	Programmer	Ligne 4 : `return True` ligne 6 : `return False` ligne 7 : `return False`
13	1	Programmation objet	`adresse_ip` est un attribut. `est_vide` est une méthode.

Question	*Niveau*	*Contenu*	*Solution*
14	2	Programmation	`def est_vide(self):` `  return self.adresse_ip == ''`
15	1	Arbre binaire de recherche	La recherche dans un arbre binaire de recherche est réputée très rapide. Le coût de l'algorithme de recherche peut être logarithmique, alors que le coût de la recherche dans un tableau est linéaire.

16	2	Programmation	`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	2	Programmation	`elif precede(ip_bin(adresse_ip), ip_bin(self.adresse_ip)):`