Ce document a été mis en ligne par l'organisme FormaV®.

Toute reproduction, représentation ou diffusion, même partielle, sans autorisation préalable, est strictement interdite.

Pour consulter tous nos sujets d'annales et corrigés du Bac Général Maths + NSI, veuillez visiter :
bac-maths-nsi.fr/annales-sujets-et-corriges

25NSIJ1ME1

Exercice 1 (6 points)

Cet exercice porte sur les bases de données relationnelles et les requêtes SQL.

Partie A

1. La clé `num_ser` n'est pas unique (p.ex les guitares d'`id = 4` et d'`id=8` ont la même valeur pour `num_ser`).
2. Le résultat de `SELECT marque, modele FROM inventaire WHERE annee = 1956;` sur l'extrait de table est :

marque	modele
Gibson	Les Paul Goldtop
Fender	Stratocaster

3. La requête `SELECT annee FROM inventaire WHERE modele = "Les Paul Standard";` convient.
4. La requête `SELECT DISTINCT modele FROM inventaire WHERE marque = "Gibson" ORDER BY annee;` convient.
5. La requête `UPDATE inventaire SET annee = 1957 WHERE id = 1;` convient.

Partie B

6. Les tables doivent être créées dans un ordre qui respecte les contraintes de référence, en commençant par les tables qui sont référencées par d'autres tables. Par exemple l'ordre `marque`, `modele`, `guitare` convient.
7. La requête `SELECT num_ser, annee FROM guitare JOIN modele ON guitare.id_modele = modele.id WHERE modele.nom = "Les Paul Standard";` convient.
8. La requête `DELETE FROM guitare WHERE id = 3;` convient.
9. La suite de requêtes ci-dessous convient et respecte les contraintes de référence.
```
INSERT INTO marque VALUES (3,"BC Rich");
INSERT INTO modele VALUES (5, "Mockingbird", 3);
INSERT INTO guitare VALUES (9, 5, 1992, "92R", 5000);
```
10. La requête `SELECT SUM(prix) FROM guitare JOIN modele ON guitare.id_modele = modele.id WHERE modele.nom = "Stratocaster";` convient.

Exercice 2 (6 points)

Cet exercice porte sur l'algorithmique, les structures de données, et la gestion de processus.

1. Le code ci-dessous permet d'initialiser les tâches `tache1` et `tache2`

```
tache1 = Tache(1, "Répondre aux e-mails", 45)
tache2 = Tache(2, "Ranger ma chambre", 60)
```

2. On complète le code de la méthode `avancer`.

```
    def avancer(self, n):
        self.duree_restante = self.duree_restante - n
```

3. On complète le code de la méthode `est_terminee`.

```
    def est_terminee(self):
        return self.duree_restante <= 0
```

4. Avec la représentation de l'énoncé, et en partant de la file proposée, on obtient après insertion :
`[debut] (<t3>, 4) (<t7>, 4) (<t1>, 3) (<t2>, 3) (<t6>, 2) (<t4>, 1) (<t5>, 1) [fin]`
5. En partant de `[début] (<t3>, 4) (<t1>, 3) (<t2>, 3) (<t4>, 1) (<t5>, 1)[fin]`, la valeur de `f.defiler()[0]` est `<t3>` et la file devient `[début] (<t1>, 3) (<t2>, 3) (<t4>, 1) (<t5>, 1) [fin]`.
6. En repartant de la file `[début] (<t3>, 4) (<t1>, 3) (<t2>, 3) (<t4>, 1) (<t5>, 1) [fin]`, la valeur de `f.examiner()[1]` est `4` et la file est inchangée.

7. On complète le code de la fonction `ajouter_file_prio`.

```
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. En comptant en nombre d'opérations sur les files (`examiner`, `est_vide`, `defiler`, `enfiler`), les deux premières boucles `while` nécessitent $4k + 3(m - k) \leqslant 4m$ opérations, où $k \leqslant m$ est le nombre de tours de la première boucle, et la dernière boucle `while` nécessite $3(m + 1)$ opérations, donc en tout dans le pire des cas $7m + 4$ opérations avec la ligne 5. C'est donc un coût d'exécution temporel **linéaire**.

9. Les tâches seront exécutées selon les blocs :

3	7	3	3	3	1	2	1	2	2	6	6	6	4	5	4	5

10. On écrit le code d'une fonction `planning`.

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

Exercice 3 (8 points)

Cet exercice porte sur l'architecture matérielle (réseau), les arbres binaires de recherche et la programmation Python.

Partie A

1. Les adresses IP valides pour installer une nouvelle borne dans le café 1 sont `192.168.20.2` et `192.168.20.157`, car `192.168.24.10` n'est pas sur le bon réseau `192.168.20.0/24` et `192.168.20.261` n'existe pas, l'entier 261 ne tient pas dans un octet.
2. L'adresse de diffusion (broadcast) du réseau du café 1 est `192.168.20.255`.
3. L'adresse `192.168.20.0` est l'adresse du réseau, l'adresse `192.168.20.255` est celle du broadcast, les adresses utilisables sont donc les 254 adresses de `192.168.20.1` à `192.168.20.254`, donc il est possible de connecter encore 250 machines puisque 4 sont déjà connectées, le routeur et les trois bornes de commande.
4. Si l'adresse machine se contente de trois bits, on peut utiliser vingt-neuf bits pour l'adresse réseau, ce qui donne un masque égal à 11111111.11111111.11111111.11111000 en binaire, FFFFFFF8 en hexadécimal, 255.255.255.248 en décimal, et /29 en CIDR.

Partie B

5. On complète la table avec les deux dernières lignes.

Routeur 2			
Réseau destination	Interface de 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. On pourrait atteindre le réseau `172.16.0.0` en passant par le routeur 1 plutôt que par le routeur 3, en transformant la ligne

Routeur 2			
Réseau destination	Interface de sortie	Prochain routeur	Nombre de sauts
172.16.0.0	172.16.4.1	172.16.4.2	1

en

Routeur 2			
Réseau destination	Interface de sortie	Prochain routeur	Nombre de sauts
172.16.0.0	172.16.3.1	172.16.3.2	1

ou bien le réseau `192.168.10.0` en passant par le routeur 3 plutôt que par le routeur 1, en transformant la ligne

Routeur 2			
Réseau destination	Interface de sortie	Prochain routeur	Nombre de sauts
192.168.10.0	172.16.3.1	172.16.3.2	2

en

Routeur 2			
Réseau destination	Interface de sortie	Prochain routeur	Nombre de sauts
172.16.0.0	172.16.4.1	172.16.4.2	2

7. On ajoute la ligne demandée.

Routeur 2			
Réseau destination	Interface de sortie	Prochain routeur	Nombre de sauts
autre	172.16.3.1	172.16.3.2	

Partie C

8. On complète le tableau des coûts.

Tableau des coûts		
Type de connexion	Débit en bit.s^{-1}	coût
Ethernet	10 Mbit.s^{-1} = 10^7 bit.s^{-1}	100
Fast Ethernet	100 Mbit.s^{-1} = 10^8 bit.s^{-1}	10
Ethernet	1 Gbit.s^{-1} = 10^9 bit.s^{-1}	1

9. Le chemin le plus court du routeur 1 au routeur 4 est 1-2-3-4 et son coût OSPF est $10 + 10 + 1 = 21$.

Partie D

10. `ip_bin('192.168.20.12')` renvoie la chaîne `'11000000.10101000.00010100.00001100'`.
11. L'instruction `return` de la ligne 7 est exécutée lorsque les deux adresses sont identiques.
12. On complète les lignes demandées.

```
1  def precede(ip_1, ip_2):
2      for i in range(35):
3          if ip_1[i] < ip_2[i]:
4              return True
5          elif ip_1[i] > ip_2[i]:
6              return False
7      return False
```

13. Les attributs de `Abr` sont `adresse_ip`, `interface`, `passerelle`, `cout`,`gauche`, `droite`, la méthode est `est_vide` (mais d'autres méthodes sont ajoutées ensuite).

14. On complète la ligne 14.

```
14          return self.adresse_ip == ''
```

15. L'utilisation d'un ABR permet une recherche rapide dans la table de routage, en temps logarithmique. On aurait la même complexité avec une recherche dichotomique dans une table de routage triée ; par contre une recherche linéaire dans la table de routage possède une complexité moins bonne, prend plus de temps en moyenne.

16. On factorise le code de la méthode `modifie`.

```
16  def modifie(self, adresse_ip,
17              interface, passerelle, cout):
18
19      if self.est_vide():
20          self.gauche = Abr('','','',0)
21          self.droite = Abr('','','',0)
22      self.adresse_ip = adresse_ip
23      self.interface = interface
24      self.passerelle = passerelle
25      self.cout = cout
```

17. On complète la ligne 35.

```
35          elif precede(adresse_ip, self.adresse_ip):
```

ou bien, si les adresses IP sont stockées en décimal pointé (l'énoncé n'est pas clair sur ce point),

```
35          elif precede(ip_bin(adresse_ip), ip_bin(self.adresse_ip)):
```

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.