Bac Général Maths + NSI (Mathématiques et Numérique et sciences informatiques)

CORRIGÉ NON OFFICIEL

Proposition de corrigé rédigée par FormaV avec l'aide d'outils d'intelligence artificielle. Elle peut contenir des erreurs et ne remplace ni le corrigé ni le barème officiels de l'examen.

Les sujets officiels des sessions passées et nos sujets d'entraînement sont classés sur https://bac-maths-nsi.fr/annales-sujets-et-corriges

Document mis en ligne par FormaV®. Reproduction ou diffusion sans autorisation interdite.

BACCALAURÉAT GÉNÉRAL • CORRIGÉ DÉTAILLÉ

Numérique et sciences informatiques

Session 2022 • Amérique du Sud • Jour 1

Code sujet **22-NSIJ1AS1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Bases de données relationnelles et requêtes SQL
9 réponses • sujet p. 2–3

Exercice 2 — Programmation et algorithmes de tri
13 réponses • sujet p. 4–5

Exercice 3 — Arbres binaires et arbre de décision
8 réponses • sujet p. 6–9

Exercice 4 — Gestion des processus et des ressources par un système d’exploitation
6 réponses • sujet p. 10–11

Exercice 5 — Réseaux et protocoles de routage
10 réponses • sujet p. 12–14

Les numéros, parties et pages renvoient au sujet original. Chaque réponse rappelle la consigne, explique la démarche et donne le résultat.

FormaV - document d'entraînement non officiel

Exercice 1

4 points

Bases de données relationnelles et requêtes SQL

Question 1.a

Sujet p. 2, 3

> Indiquer si idMere peut être ou non une clé primaire de la relation Naissances. Justifier la réponse.

Une clé primaire doit identifier chaque enregistrement de façon unique. Or `idMere` identifie la patiente, pas un bébé unique : une même patiente peut avoir plusieurs naissances enregistrées. Dans l'extrait, la patiente numéro 13861 a deux bébés enregistrés.

> **RÉPONSE À RETENIR**
>
> `idMere` ne peut pas être une clé primaire de `Naissances`, car sa valeur peut se répéter.

Question 1.b

Sujet p. 2, 3

> Indiquer si le couple (date, rang) peut être ou non une clé primaire de la relation Naissances. Justifier la réponse.

Le rang recommence à 1 au début de chaque mois et distingue les naissances d'un même mois. Si deux enregistrements ont la même date, leurs rangs sont donc différents. Si leurs dates diffèrent, le couple diffère déjà par sa date.

> **RÉPONSE À RETENIR**
>
> Le couple `(date, rang)` peut être une clé primaire de `Naissances` : il identifie une naissance de façon unique selon la règle donnée pour le rang mensuel.

Question 1.c

Sujet p. 2, 3

> Indiquer si le couple (poids, taille) peut être ou non une clé primaire de la relation Naissances. Justifier la réponse.

Une clé primaire doit être garantie unique pour tous les enregistrements. Rien dans le schéma ni dans les règles données n'empêche deux bébés d'avoir le même poids et la même taille. Le fait que les couples visibles dans l'extrait soient différents ne garantit pas cette unicité dans la table complète.

> **RÉPONSE À RETENIR**
>
> Le couple `(poids, taille)` ne peut pas être retenu comme clé primaire : son unicité n'est pas garantie.

Question 2.

Sujet p. 2, 3

> Pourquoi la requête DELETE FROM Patientes WHERE numPatiente = 13858; provoque-t-elle une erreur ?

La valeur 13858 apparaît dans `Naissances.idMere`, qui est une clé étrangère vers `Patientes.numPatiente`. Supprimer la patiente correspondante laisserait une naissance référencer un numéro de patiente inexistant. Aucune règle de suppression en cascade n'étant indiquée, la contrainte de clé étrangère interdit cette suppression.

RÉPONSE À RETENIR

La suppression provoque une erreur de contrainte d'intégrité référentielle : la patiente numéro 13858 est référencée par au moins une ligne de `Naissances`.

Question 3.

Sujet p. 2, 3

Donner en langage SQL la requête d'insertion qui permet d'inscrire, avec le numéro 13862, la patiente dénommée Ninette Bélanger résidant à La Rochelle.

On insère une ligne dans `Patientes` en fournissant les quatre attributs de la relation. Le nom est Bélanger et le prénom est Ninette.

RÉPONSE À RETENIR

```
INSERT INTO Patientes (numPatiente, nom, prenom, commune)
VALUES (13862, 'Bélanger', 'Ninette', 'La Rochelle');
```

Question 4.

Sujet p. 2, 3

Une erreur a été commise lors de la saisie du prénom du bébé de madame Fernandez. La bonne orthographe est Laurette. Donner en langage SQL l'instruction à exécuter pour corriger le prénom du bébé.

Il faut modifier l'attribut `prenom` dans `Naissances`. La condition utilise le numéro de la mère Fernandez (13860) et le prénom erroné figurant dans l'extrait afin de cibler le bébé concerné.

RÉPONSE À RETENIR

```
UPDATE Naissances
SET prenom = 'Laurette'
WHERE idMere = 13860 AND prenom = 'Lorette';
```

Question 5.

Sujet p. 2, 3

Écrire une requête SQL qui renvoie la liste des patientes (attributs nom et prenom) dont la commune de résidence est Aigrefeuille d'Aunis.

Les informations demandées sont dans `Patientes`. On sélectionne seulement `nom` et `prenom`, puis on filtre sur la commune exacte.

RÉPONSE À RETENIR

Requête :

```
SELECT nom, prenom
FROM Patientes
WHERE commune = 'Aigrefeuille d'Aunis';
```

Résultat sur l'extrait fourni :

nom	prenom
Berthelot	Michelle

Question 6.

Sujet p. 2, 3

> Écrire une requête SQL qui renvoie le poids moyen des bébés nés par césarienne.

On relie chaque naissance à son type d'accouchement par la clé étrangère `acc` et la clé primaire `idAcc`. On filtre sur le libellé « césarienne » et on calcule la moyenne de `poids` pour les lignes correspondantes.

RÉPONSE À RETENIR

Requête :

```
SELECT AVG(Naissances.poids)
FROM Naissances
JOIN TypesAccouchement
ON Naissances.acc = TypesAccouchement.idAcc
WHERE TypesAccouchement.libelleAcc = 'césarienne';
```

Le résultat numérique n'est pas déterminable à partir des extraits fournis : l'extrait de `TypesAccouchement` ne donne pas les valeurs de `idAcc`, et celui de `Naissances` ne montre qu'une partie des lignes. La requête calcule la moyenne sur les données complètes de la base.

Question 7.

Sujet p. 2, 3

> Quels sont les enregistrements renvoyés par la requête suivante lancée sur les extraits donnés des tables ? La requête sélectionne Patientes.nom et Patientes.prenom, joint Naissances à TypesAccouchement avec Naissances.acc = TypesAccouchement.idAcc, joint Naissances à Patientes avec Naissances.idMere = Patientes.numPatiente, puis filtre sur TypesAccouchement.idAcc = 1.

La jointure associe chaque naissance à sa patiente et à son type d'accouchement. Le filtre conserve les naissances dont `acc` vaut 1. Dans l'extrait, elles concernent Berthelot, Samson et les deux bébés de Baugé. Comme la requête ne contient pas `DISTINCT`, la ligne de Baugé apparaît deux fois : chaque naissance produit un enregistrement.

RÉPONSE À RETENIR

nom	prenom
Berthelot	Michelle
Samson	Marine
Baugé	Gaëlle
Baugé	Gaëlle

FormaV - document d'entraînement non officiel

Exercice 2

4 points

Programmation et algorithmes de tri

Question 1.

Sujet p. 4

> Écrire l'instruction qui permet d'insérer dans la liste attente, définie par attente = [(45,2),(47,1),(49,3)], un nouveau patient identifié 50 avec une priorité de 4.

Chaque patient est représenté par un tuple composé de son identifiant et de sa priorité. La méthode `append` ajoute le tuple du nouveau patient à la liste `attente`. La liste donnée n'est pas triée par priorité, donc on ne peut pas dire que l'ajout à la fin la rend ordonnée.

RÉPONSE À RETENIR

```
attente.append((50, 4))
```

La liste devient `[(45, 2), (47, 1), (49, 3), (50, 4)]`.

Question 2.a (option 1)

Sujet p. 4

> Quel algorithme de tri est implémenté par la fonction tri(attente) donnée ? Option : le tri fusion.

À chaque tour, le code cherche la plus petite priorité dans la partie de la liste allant de l'indice `i` à la fin, puis échange l'élément trouvé avec celui d'indice `i`. C'est le principe du tri par sélection, et non du tri fusion.

RÉPONSE À RETENIR

Option incorrecte : le tri fusion. La réponse est le tri par sélection.

Question 2.a (option 2)

Sujet p. 4

> Quel algorithme de tri est implémenté par la fonction tri(attente) donnée ? Option : le tri par insertion.

À chaque tour, le code cherche la plus petite priorité dans la partie de la liste allant de l'indice `i` à la fin, puis échange l'élément trouvé avec celui d'indice `i`. Il ne construit pas une partie triée en insérant successivement chaque élément : il réalise un tri par sélection.

RÉPONSE À RETENIR

Option incorrecte : le tri par insertion. La réponse est le tri par sélection.

Question 2.a (option 3)

Sujet p. 4

> Quel algorithme de tri est implémenté par la fonction tri(attente) donnée ? Option : le tri par sélection.

Pour chaque position `i`, le programme recherche l'indice `pos` du minimum de la partie non encore triée, puis échange les éléments d'indices `i` et `pos`. C'est précisément le tri par sélection.

RÉPONSE À RETENIR

Option correcte : le tri par sélection.

FormaV - document d'entraînement non officiel

Question 2.a (option 4)

Sujet p. 4

> Quel algorithme de tri est implémenté par la fonction tri(attente) donnée ? Option : le tri rapide.

Le code ne choisit pas de pivot et ne partitionne pas la liste en deux parties. Il recherche plutôt le minimum de la partie restante à chaque étape : c'est le tri par sélection.

> **RÉPONSE À RETENIR**
>
> Option incorrecte : le tri rapide. La réponse est le tri par sélection.

Question 2.b (option 1)

Sujet p. 4

> Quelle est la complexité en temps des tris par insertion et par sélection ? Option : constante, O(1).

Le tri par sélection effectue un nombre quadratique de comparaisons. Le tri par insertion peut aussi nécessiter un nombre quadratique d'opérations dans le pire cas. Son meilleur cas est toutefois linéaire, par exemple lorsque la liste est déjà triée.

> **RÉPONSE À RETENIR**
>
> Option incorrecte : constante, $O(1)$. La réponse attendue est quadratique, $O(n^2)$ dans le pire cas.

Question 2.b (option 2)

Sujet p. 4

> Quelle est la complexité en temps des tris par insertion et par sélection ? Option : logarithmique, O(log n).

Les tris par insertion et par sélection ne se limitent pas à un nombre logarithmique d'opérations. Dans le pire cas, leur complexité est quadratique : $O(n^2)$.

> **RÉPONSE À RETENIR**
>
> Option incorrecte : logarithmique, $O(\log n)$. La réponse attendue est quadratique, $O(n^2)$ dans le pire cas.

Question 2.b (option 3)

Sujet p. 4

> Quelle est la complexité en temps des tris par insertion et par sélection ? Option : linéaire, O(n).

Le tri par sélection examine successivement des parties de tailles n, $n-1$, etc. Il effectue donc un nombre quadratique de comparaisons. Le tri par insertion peut également nécessiter un nombre quadratique d'opérations dans le pire cas, même si son meilleur cas est linéaire.

> **RÉPONSE À RETENIR**
>
> Option incorrecte : linéaire, $O(n)$. La réponse attendue est quadratique, $O(n^2)$ dans le pire cas.

Question 2.b (option 4)

Sujet p. 4

> Quelle est la complexité en temps des tris par insertion et par sélection ? Option : quadratique, O(n²).

Au tri par sélection, les recherches successives du minimum examinent des parties de tailles n, $n-1$, jusqu'à 1. Le total est proportionnel à n^2. Le tri par insertion a également une complexité quadratique dans le pire cas, même si son meilleur cas est linéaire.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Option correcte : quadratique, $O(n^2)$ dans le pire cas.

Question 3.a

Sujet p. 5

> On considère que la salle d'attente est triée par ordre croissant des priorités. La fonction quitte(attente) supprime le patient de priorité 1 de la liste attente passée en paramètre et renvoie la nouvelle salle d'attente. Recopier et compléter la ligne 14 en définissant une liste en compréhension.

La liste étant triée par ordre croissant des priorités et les priorités étant distinctes, le patient de priorité 1 est le premier élément. La compréhension ci-dessous conserve les tuples dont la priorité n'est pas 1 ; elle renvoie donc une nouvelle liste sans ce patient.

RÉPONSE À RETENIR

```
def quitte(attente):
    return [(patient, prio) for patient, prio in attente if prio != 1]
```

Par exemple, `quitte([(47, 1), (45, 2), (49, 3)])` renvoie `[(45, 2), (49, 3)]`.

Question 3.b

Sujet p. 5

> La fonction quitte(attente) ne met pas à jour les priorités des patients. Écrire une fonction maj(attente) qui met à jour, dans une nouvelle liste, les priorités des patients à la suite de la prise en charge d'un patient et renvoie cette nouvelle salle d'attente. Exemple : maj([(45, 2), (49, 3)]) renvoie [(45, 1), (49, 2)].

Après la prise en charge du patient de priorité 1, cette fonction reçoit la liste des patients restants. Leurs priorités diminuent alors toutes de 1 : l'ancienne priorité 2 devient 1, l'ancienne priorité 3 devient 2, etc. La compréhension crée une nouvelle liste de tuples.

RÉPONSE À RETENIR

```
def maj(attente):
    return [(patient, prio - 1) for patient, prio in attente]
```

Ainsi, `maj([(45, 2), (49, 3)])` renvoie `[(45, 1), (49, 2)]`.

Question 4.a

Sujet p. 5

> Écrire la fonction priorite(attente, p) qui renvoie l'ordre de priorité d'un patient p de la liste attente. Exemple : priorite([(45,2),(47,1),(49,3)], 49) renvoie 3.

On parcourt les tuples de la liste jusqu'à trouver celui dont l'identifiant correspond à `p`. La seconde composante de ce tuple est sa priorité ; on la renvoie. L'exemple suppose que le patient recherché est bien présent dans la liste.

RÉPONSE À RETENIR

```
def priorite(attente, p):
    for patient, prio in attente:
        if patient == p:
            return prio
```

Ainsi, `priorite([(45, 2), (47, 1), (49, 3)], 49)` renvoie `3`.

Question 4.b

Sujet p. 5

La fonction revise(attente, p) renvoie une nouvelle salle d'attente non triée où le patient p dont la priorité médicale a évolué reçoit la priorité 1 et où les priorités des autres patients sont mises à jour. Recopier et compléter les lignes 28 et 29 du code fourni. Exemple : revise([(45,2),(47,1),(49,3)], 49) renvoie [(45,3),(47,2),(49,1)].

La variable `n` contient l'ancienne priorité du patient `p`. Ce patient reçoit la priorité 1. Pour les autres patients dont la priorité est strictement inférieure à `n`, on augmente la priorité numérique de 1 afin de libérer le rang devenu prioritaire. Les priorités des autres patients restent inchangées. Les lignes à compléter sont donc l'ajout du tuple de `p`, puis le test `prio < n`.

RÉPONSE À RETENIR

```
def revise(attente, p):
    nouvelle = []
    n = priorite(attente, p)
    for patient, prio in attente:
        if patient == p:
            nouvelle.append((patient, 1))
        elif prio < n:
            nouvelle.append((patient, prio + 1))
        else:
            nouvelle.append((patient, prio))
    return nouvelle
```

Pour l'exemple fourni, `revise([(45, 2), (47, 1), (49, 3)], 49)` renvoie `[(45, 3), (47, 2), (49, 1)]`.

Exercice 3

4 points

Arbres binaires et arbre de décision

Question 1.

Sujet p. 6, 7

> Déterminer la taille et la hauteur de l'arbre de décision donné en exemple en introduction, en présence d'une jaunisse.

La taille est le nombre total de nœuds. Le schéma contient les nœuds suivants : conjonctive jaune, hypercaroténémie, bilirubine, splénomégalie, hépatomégalie, maladie de Gilbert par défaut de glycuro-conjugaison, anémie hémolytique, hépatite, douleur et fièvre, cancer de la tête du pancréas et lithiase du cholédoque. Il y a donc 11 nœuds.

La hauteur est le nombre de nœuds du plus long chemin de la racine à une feuille, conformément à la convention donnée. Un plus long chemin est conjonctive jaune → bilirubine → hépatomégalie → douleur et fièvre → cancer de la tête du pancréas : il comporte 5 nœuds.

> RÉPONSE À RETENIR
>
> **Taille : 11 nœuds. Hauteur : 5.**

Question 2.a (option 1)

Sujet p. 7

> À quelle représentation graphique correspond la structure de l'arbre implémentée par le dictionnaire fourni ? Option : arbre 1.

Dans le dictionnaire, `b` est le sous-arbre gauche de `a` et `f` son sous-arbre droit. Pour `b`, le sous-arbre gauche est vide et le sous-arbre droit est `d` : `d` doit donc se trouver à droite de `b`. Pour `f`, le sous-arbre gauche est `g` et le sous-arbre droit est vide : `g` doit se trouver à gauche de `f`. Dans l'arbre 1, `d` est à gauche de `b`, ce qui ne correspond pas au dictionnaire.

> RÉPONSE À RETENIR
>
> **L'arbre 1 ne correspond pas au dictionnaire. La représentation correcte est l'arbre 2.**

Question 2.a (option 2)

Sujet p. 7

> À quelle représentation graphique correspond la structure de l'arbre implémentée par le dictionnaire fourni ? Option : arbre 2.

Le dictionnaire indique que `b` est le fils gauche de `a`, et que `f` est son fils droit. Il indique également que `d` est le fils droit de `b` et que `g` est le fils gauche de `f`. C'est la disposition de l'arbre 2.

> RÉPONSE À RETENIR
>
> **La structure correspond à l'arbre 2.**

Question 2.a (option 3)

Sujet p. 7

> À quelle représentation graphique correspond la structure de l'arbre implémentée par le dictionnaire fourni ? Option

FormaV - document d'entraînement non officiel

: arbre 3.

Le dictionnaire place `d` dans le sous-arbre droit de `b` et `g` dans le sous-arbre gauche de `f`. Dans l'arbre 3, `g` est à droite de `f` : cette représentation ne correspond donc pas au dictionnaire.

RÉPONSE À RETENIR

L'arbre 3 ne correspond pas au dictionnaire. La représentation correcte est l'arbre 2.

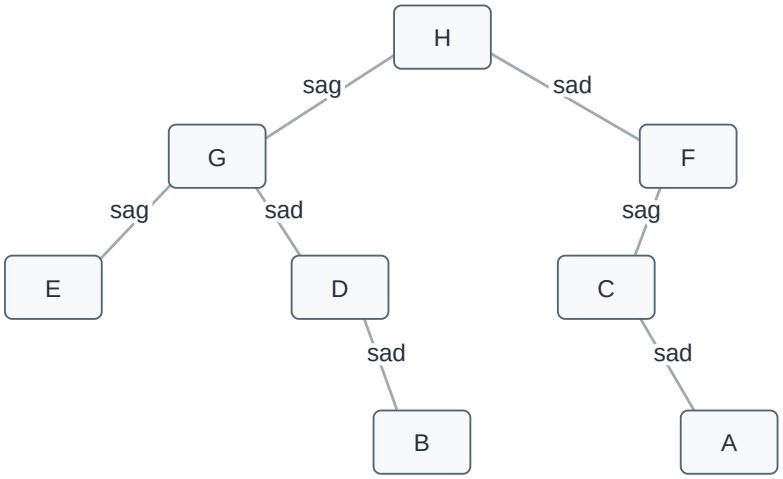
Question 2.b

Sujet p. 8

Représenter graphiquement l'arbre correspondant au code fourni, dont la racine est H.

La clé `'sag'` représente le sous-arbre gauche et `'sad'` le sous-arbre droit. On lit donc : `H` a pour fils gauche `G` et pour fils droit `F` ; `G` a pour fils gauche `E` et pour fils droit `D` ; `D` a pour fils droit `B` ; `F` a pour fils gauche `C` ; et `C` a pour fils droit `A`. Les sous-arbres associés à `{}` sont vides.

Arbre binaire de racine H



Les arêtes représentent les sous-arbres non vides ; sag désigne le fils gauche et sad le fils droit.

RÉPONSE À RETENIR

Voici l'arbre binaire correspondant au dictionnaire fourni.

Question 3.a

Sujet p. 8

Donner l'affichage après l'appel de la fonction parcours avec l'arbre dont la représentation graphique est donnée sur la page.

La fonction appelle d'abord le parcours du sous-arbre gauche, puis celui du sous-arbre droit, et affiche ensuite l'étiquette du nœud courant. Elle affiche donc d'abord `d`, puis `b`, ensuite `g`, puis `f`, et enfin la racine `a`.

RÉPONSE À RETENIR

L'affichage, une étiquette par ligne, est :

```
d
b
g
f
a
```

Question 3.b

Sujet p. 8

> Écrire une fonction parcours_maladies(arb) qui n'affiche que les feuilles de l'arbre binaire non vide arb passé en argument, ce qui correspond aux maladies possiblement induites par l'arbre de décision.

Une feuille est un nœud dont les deux sous-arbres sont vides. La fonction parcourt les sous-arbres gauche et droit, puis affiche l'étiquette du nœud seulement si les deux sont vides. Le test de l'arbre vide arrête la récursion dans les sous-arbres vides.

Par exemple, appliquée à l'arbre de la question 3.a, cette fonction affiche `d` et `g`, les seules feuilles de cet arbre.

RÉPONSE À RETENIR

```
def parcours_maladies(arb):
    if arb == {}:
        return None
    parcours_maladies(arb['sag'])
    parcours_maladies(arb['sad'])
    if arb['sag'] == {} and arb['sad'] == {}:
        print(arb['etiquette'])
```

Question 4.

Sujet p. 6, 9

> Compléter les lignes 6, 8, 14 et 18 du code de la fonction symptomes(arb, mal), qui affiche l'ensemble des symptômes relatifs à une maladie. La clé surChemin, booléenne, est initialisée à False pour tous les nœuds. L'exemple fourni concerne la maladie « anémie hémolytique ».

La fonction explore récursivement les sous-arbres avant de traiter le nœud courant. Quand l'étiquette du nœud est la maladie recherchée, elle marque ce nœud comme étant sur le chemin et affiche l'en-tête. En remontant vers la racine, si le chemin passe par le sous-arbre droit, le symptôme est présent : on affiche son étiquette. S'il passe par le sous-arbre gauche, le symptôme est absent : on affiche « pas de » suivi de son étiquette.

Pour l'anémie hémolytique, le chemin remonte par le fils droit de `splénomégalie`, le fils gauche de `bilirubine`, puis le fils droit de `conjonctive jaune`. Cela conduit à afficher, dans cet ordre, « splénomégalie », « pas de bilirubine » et « conjonctive jaune » après l'en-tête de la maladie.

RÉPONSE À RETENIR

Les lignes à compléter sont :

```
06        symptomes(arb['sad'], mal)
08    if arb['etiquette'] == mal:
14            print(arb['etiquette'])
18            print('pas de', arb['etiquette'])
```

La fonction complétée est :

```
def symptomes(arb, mal):
    if arb['sag'] != {}:
        symptomes(arb['sag'], mal)

    if arb['sad'] != {}:
        symptomes(arb['sad'], mal)

    if arb['etiquette'] == mal:
        arb['surChemin'] = True
        print('symptômes de', arb['etiquette'], ':')

    else:
        if arb['sad'] != {} and arb['sad']['surChemin']:
            print(arb['etiquette'])
            arb['surChemin'] = True

        if arb['sag'] != {} and arb['sag']['surChemin']:
            print('pas de', arb['etiquette'])
            arb['surChemin'] = True
```

Exercice 4

4 points

Gestion des processus et des ressources par un système d'exploitation

A. Ordonnancement des processus

Question 1.

Sujet p. 10

> Reproduire le diagramme et indiquer dans chacune des cases le processus exécuté par le processeur entre deux unités de temps, en appliquant l'ordonnancement préemptif décrit et les données du tableau pour les processus P1 à P4. Il peut y avoir des cases vides.

On examine les arrivées et les priorités à chaque unité de temps. Parmi les processus déjà arrivés et non terminés, le processeur exécute celui dont le numéro de priorité est le plus petit. Un processus préempté reprend ensuite avec le temps d'exécution qui lui reste.

- De 0 à 2, seul P1 est arrivé : il exécute 2 unités.
- À 2, P2 arrive et préempte P1, car son numéro de priorité 2 est inférieur à 4. P2 exécute de 2 à 3.
- À 3, P3 arrive avec la priorité 1 et préempte P2. Il exécute ses 3 unités de 3 à 6. L'arrivée de P4 à 5 ne le préempte pas.
- De 6 à 9, P2 exécute ses 3 unités restantes et termine. P4 est alors prioritaire sur P1.
- P4 exécute de 9 à 13, puis P1 termine son unité restante de 13 à 14. Aucun processus ne reste à exécuter de 14 à 15.

RÉPONSE À RETENIR

Les processus dans les cases successives du diagramme, de l'intervalle [0,1[à l'intervalle [14,15[, sont :

[0,1[	**[1,2[**	**[2,3[**	**[3,4[**	**[4,5[**	**[5,6[**	**[6,7[**	**[7,8[**	**[8,9[**	**[9,10[**	**[10,11[**	**[11,12[**	**[12,13[**	**[13,14[**	**[14,15[**
P1	P1	P2	P3	P3	P3	P2	P2	P2	P4	P4	P4	P4	P1	Vide

Question 2.

Sujet p. 10

> Recopier et compléter les temps de séjour et les temps d'attente de chacun des processus, en unités de temps. Temps de séjour = instant de terminaison − instant d'arrivée ; temps d'attente = temps de séjour − temps d'exécution.

D'après l'ordonnancement, les instants de terminaison sont : P1 à 14, P2 à 9, P3 à 6 et P4 à 13. On applique ensuite les formules données : temps de séjour = instant de terminaison − instant d'arrivée, puis temps d'attente = temps de séjour − temps d'exécution.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Processus	Temps d'exécution	Instant d'arrivée	Temps de séjour	Temps d'attente
P1	3	0	14 − 0 = 14	14 − 3 = 11
P2	4	2	9 − 2 = 7	7 − 4 = 3
P3	3	3	6 − 3 = 3	3 − 3 = 0
P4	4	5	13 − 5 = 8	8 − 4 = 4

Question 3.

Sujet p. 10

> À quelles conditions le temps d'attente d'un processus peut-il être nul ?

Le temps d'attente est le temps passé entre l'arrivée et la terminaison sans utiliser le processeur. Il est nul lorsque le processus ne connaît aucune période d'attente pendant cet intervalle.

RÉPONSE À RETENIR

Le temps d'attente est nul si le processus commence à être exécuté dès son arrivée et s'exécute ensuite sans interruption jusqu'à sa terminaison.

B. Processus et ressources

Question 1.

Sujet p. 11

> À partir du tableau des ressources D1 à D5, démontrer qu'à cet instant les processus s'attendent mutuellement.

Pour chaque ressource notée A, on repère le processus qui la demande et celui qui la mobilise (M). Le tableau indique que l'analyseur attend D4, que le SGBD mobilise ; le SGBD attend D5, que le tableur mobilise ; le tableur attend D1, que l'analyseur mobilise ; et le traitement de texte attend D3, que le tableur mobilise.

RÉPONSE À RETENIR

On obtient notamment le cycle d'attente suivant : **analyseur d'échantillon → SGBD → tableur → analyseur d'échantillon**. En effet, l'analyseur attend D4 détenue par le SGBD, le SGBD attend D5 détenue par le tableur, et le tableur attend D1 détenue par l'analyseur. Le traitement de texte attend lui aussi une ressource, D3, détenue par le tableur ; il est bloqué, même s'il ne fait pas partie de ce cycle.

Question 2.

Sujet p. 11

> Comment s'appelle cette situation dans laquelle les processus s'attendent mutuellement ?

Le cycle établi à la question précédente montre que des processus attendent des ressources détenues par d'autres processus du cycle, qui ne peuvent donc pas progresser.

RÉPONSE À RETENIR

Cette situation s'appelle un **interblocage** (ou *deadlock*).

FormaV - document d'entraînement non officiel

Question 3.

Sujet p. 11

> On suppose que l’analyseur d’échantillon libère la ressource D1. Donner un ordre possible d’exécution des processus.

Après la libération de D1 par l’analyseur, le tableur peut obtenir la ressource qu’il attend. S’il termine et libère les ressources qu’il détient, le SGBD peut obtenir D5, puis terminer et libérer D4 afin que l’analyseur puisse reprendre. Le traitement de texte peut, lui, reprendre dès que le tableur libère D3. On suppose que les processus qui reprennent peuvent terminer et libérer leurs ressources.

RÉPONSE À RETENIR

Un ordre possible d’exécution est : **tableur, SGBD, analyseur d’échantillon, traitement de texte**. Le traitement de texte peut être placé après l’analyseur dans cet ordre, puisqu’une fois le tableur terminé, D3 est libérée et il n’attend plus cette ressource.

FormaV - document d'entraînement non officiel

Exercice 5

4 points

Réseaux et protocoles de routage

A. Adressage

Question 1.

Sujet p. 12, 13

> Quels sont l'adresse et le masque du réseau local du service de radiologie (RLR) ?

Le schéma indique pour le réseau local du service de radiologie l'adresse `192.168.1.0/24`. Le préfixe `/24` correspond au masque binaire composé de 24 bits à 1 suivis de 8 bits à 0, soit `255.255.255.0` en notation décimale pointée.

> **RÉPONSE À RETENIR**
>
> Adresse du réseau RLR : **192.168.1.0** ; masque : **255.255.255.0** (soit **/24**).

Question 2.

Sujet p. 12, 13

> Donner les adresses des trois interfaces du routeur R5 permettant de transmettre ou de recevoir des données.

On relève sur le schéma les adresses des interfaces de R5 sur chacun de ses trois réseaux de connexion. Sur la liaison R5–R4, R5 porte l'adresse `.1` et R4 l'adresse `.254` : il faut donc bien distinguer l'adresse de l'interface de R5 de celle de R4.

> **RÉPONSE À RETENIR**
>
> - Interface de R5 vers R1 : **175.89.50.254**.
> - Interface de R5 vers R4 : **44.197.5.1**.
> - Interface de R5 vers le réseau du serveur SP : **192.168.5.254**.

Question 3.a

Sujet p. 12, 13

> Donner la première et la dernière adresse IP pouvant être attribuées à une machine sur le réseau RLR.

Le réseau RLR est `192.168.1.0/24`. Il reste donc `32 − 24 = 8` bits pour la partie machine. L'adresse avec ces 8 bits à 0 est l'adresse du réseau (`192.168.1.0`) et celle avec ces 8 bits à 1 est l'adresse de diffusion (`192.168.1.255`) ; aucune de ces deux adresses ne peut être attribuée à une machine. Les adresses attribuables vont de `192.168.1.1` à `192.168.1.254`.

> **RÉPONSE À RETENIR**
>
> Première adresse attribuable : **192.168.1.1**. Dernière adresse attribuable : **192.168.1.254**.

Question 3.b

Sujet p. 12, 13

> En déduire le nombre de machines pouvant être connectées sur le réseau RLR.

Avec 8 bits pour la partie machine, il y a $2^8 = 256$ adresses au total. Il faut exclure l'adresse du réseau et l'adresse de diffusion, qui ne sont pas attribuables à des machines : $256 - 2 = 254$.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Le réseau RLR permet de connecter **254 machines**.

B. Étude du protocole RIP (Routing Information Protocol)

Question 1.

Sujet p. 12, 13

> Avec le protocole RIP, qui utilise la distance en nombre de sauts, le serveur SP doit transmettre des données au service de radiologie via le routeur R0 en effectuant le moins de sauts possible. Citer les routeurs parcourus par le paquet.

RIP choisit le chemin comportant le moins de sauts. Le serveur SP est relié à R5 et le réseau RLR est relié à R0. Le chemin R5–R1–R0 comporte deux sauts ; les chemins par R4 nécessitent au moins trois sauts. Les routeurs traversés, dans l'ordre de SP vers RLR, sont donc R5, R1 puis R0.

RÉPONSE À RETENIR

Route : **SP → R5 → R1 → R0 → RLR**. Les routeurs parcourus sont **R5 → R1 → R0**.

Question 2.

Sujet p. 12, 13

> À la suite d'une opération de maintenance, le routeur R1 est déconnecté et aucun paquet ne peut transiter par ce routeur. Déterminer une nouvelle route empruntée par les paquets en citant les routeurs dans l'ordre.

R1 étant déconnecté, on retire ce routeur des chemins possibles. Depuis R5, on peut passer par R4 puis atteindre R0 soit par R2, soit par R3. Chacun de ces chemins compte trois sauts : R5–R4–R2–R0 et R5–R4–R3–R0. L'énoncé demande une nouvelle route : l'un ou l'autre de ces chemins minimaux convient.

RÉPONSE À RETENIR

Une route minimale possible est **SP → R5 → R4 → R2 → R0 → RLR**. La route **SP → R5 → R4 → R3 → R0 → RLR** est également minimale.

C. Protocole OSPF (Open Shortest Path First)

Question 1.

Sujet p. 14

> Calculer le coût de la liaison entre R2 et R3, dont la bande passante est de 400 Mb/s, à l'aide de la formule donnée.

La liaison R2–R3 a une bande passante de 400 Mb/s, soit `400 000 000` b/s. La formule donne 10^9 `/ 400 000 000 = 2,5`. Comme le résultat n'est pas entier, le coût est l'entier immédiatement supérieur : 3.

RÉPONSE À RETENIR

Le coût de la liaison **R2–R3 est 3**.

FormaV - document d'entraînement non officiel

Question 2.

Sujet p. 14

> Donner une bande passante possible de la connexion entre R3 et R4, dont le coût indiqué dans le tableau est 5.

Il faut une bande passante pour laquelle l'entier immédiatement supérieur de 10^9 / BP vaut 5. Par exemple, avec 200 Mb/s, soit 200 000 000 b/s, le quotient vaut exactement 10^9 / 200 000 000 = 5. Cette valeur donne donc bien un coût de 5.

> **RÉPONSE À RETENIR**
>
> Une bande passante possible pour la liaison **R3–R4 est 200 Mb/s**.

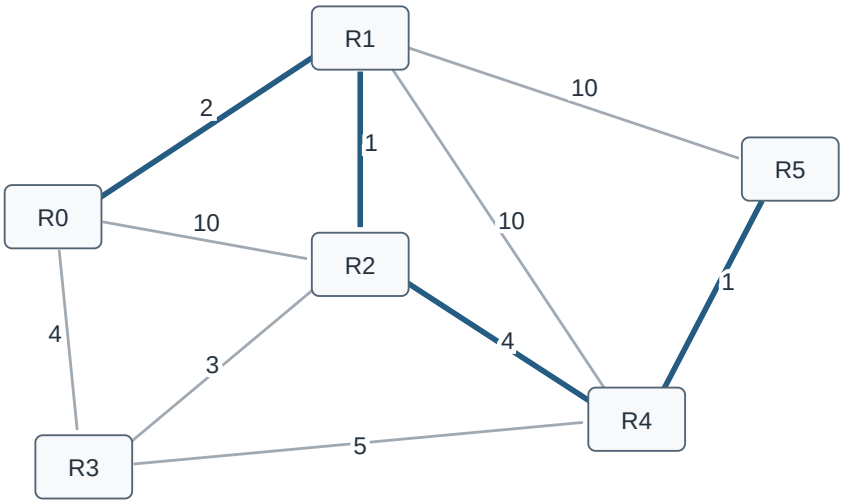
Question 3.

Sujet p. 12, 14

> Déterminer le chemin parcouru par un paquet partant du serveur patients (SP) vers le service de radiologie (RLR) en utilisant le protocole OSPF. Préciser également le coût de ce chemin.

On compare les coûts des chemins entre R5 et R0. Le chemin R5–R4–R2–R1–R0 coûte 1 + 4 + 1 + 2 = 8. Le chemin R5–R4–R3–R0 coûte 1 + 5 + 4 = 10, et le chemin R5–R1–R0 coûte 10 + 2 = 12. Les autres passages par R2 ou R1 ne donnent pas un coût inférieur à 8. Le chemin de coût minimal est donc R5–R4–R2–R1–R0. Les liaisons terminales SP–R5 et R0–RLR ne sont pas incluses dans le coût, car l'étude demandée porte sur le chemin entre R5 et R0.

Réseau pondéré et chemin OSPF minimal



Les nombres indiquent les coûts des liaisons entre routeurs. Le chemin mis en évidence est R5 → R4 → R2 → R1 → R0, de coût 8.

> **RÉPONSE À RETENIR**
>
> Chemin OSPF : **SP → R5 → R4 → R2 → R1 → R0 → RLR**. Son coût entre R5 et R0 est **8**.

Question 4.

Sujet p. 12, 14

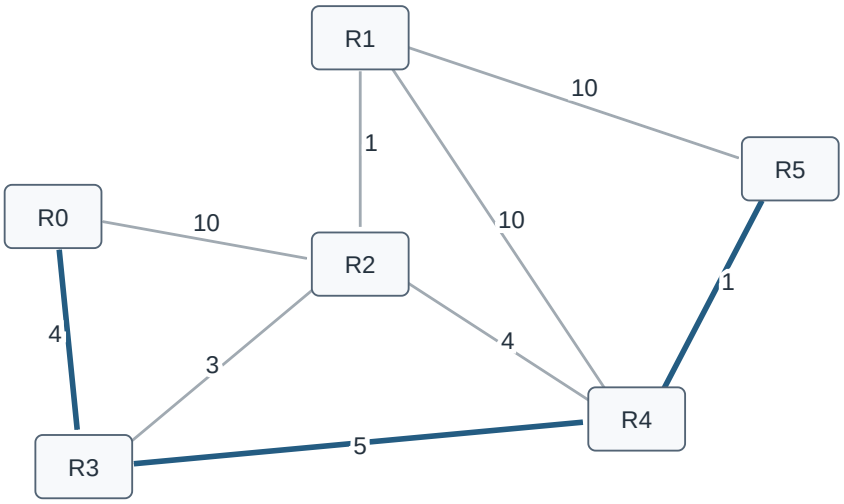
> À la suite d'une opération de maintenance, la liaison R0-R1 est déconnectée et aucun paquet ne peut transiter par

FormaV - document d'entraînement non officiel

> cette liaison. Déterminer une nouvelle route empruntée par les paquets en citant les routeurs dans l'ordre.

La liaison R0–R1 étant coupée, elle ne peut plus être utilisée. Depuis R5, le chemin R5–R4–R3–R0 coûte `1 + 5 + 4 = 10`. Le chemin R5–R4–R2–R0 coûte `1 + 4 + 10 = 15` ; passer par R2 puis R3 vers R0 coûte `1 + 4 + 3 + 4 = 12`. Le passage par R3 est donc le moins coûteux parmi les chemins possibles.

Réseau après la coupure de la liaison R0–R1



La liaison R0–R1, déconnectée, est retirée. Le chemin mis en évidence est R5 → R4 → R3 → R0, de coût 10.

> **RÉPONSE À RETENIR**
>
> Nouvelle route OSPF : **SP → R5 → R4 → R3 → R0 → RLR**. Son coût entre R5 et R0 est **10**.

Sujet : https://www.sujetdebac.fr/annales-pdf/2022/spe-numerique-informatique-2022-amerique-sud-1-sujet-officiel.pdf
© FormaV EI • Proposition de correction, distincte d'un corrigé officiel.

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.