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 (NSI)

Session 2023 • Polynésie • Jour 2

Code sujet **23-NSIJ2ME1** 3 h 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Arbres binaires de recherche, POO et récursivité
10 réponses • sujet p. 1–2

Exercice 2 — Bases de données et SQL
11 réponses • sujet p. 3–4

Exercice 3 — Architecture matérielle, processus et ordonnancement
9 réponses • sujet p. 5–7

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

Arbres binaires de recherche, POO et récursivité

1

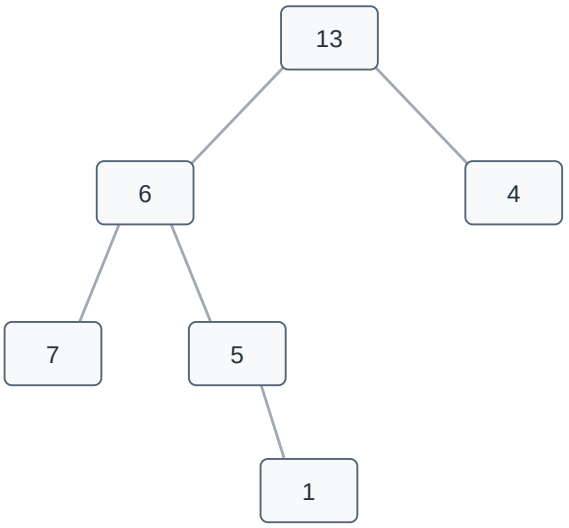
Question 1. (a)

Sujet p. 1

> On considère l'arbre représenté sur la page. Justifier que cet arbre est un arbre binaire.

Un arbre binaire est un arbre dans lequel chaque nœud possède au plus deux enfants, appelés fils gauche et fils droit. Sur le dessin, le nœud 13 a deux enfants, le nœud 6 a deux enfants, le nœud 5 a un enfant et les autres nœuds n'en ont pas plus de deux.

Arbre de la question 1



Les traits représentent les liens parent-enfant de l'arbre figurant sur la page 1.

RÉPONSE À RETENIR

Cet arbre est bien un arbre binaire : chacun de ses nœuds possède au plus deux enfants.

Question 1. (b)

Sujet p. 1

> Indiquer si l'arbre représenté sur la page est un arbre binaire de recherche (ABR). Justifier la réponse.

Dans un arbre binaire de recherche, les valeurs du sous-arbre gauche d'un nœud sont inférieures à sa valeur et celles du sous-arbre droit sont supérieures. Or le fils droit de la racine 13 vaut 4, donc est inférieur à 13 : la propriété n'est pas respectée. On relève également que le fils gauche de 6 vaut 7, supérieur à 6.

RÉPONSE À RETENIR

Non, cet arbre n'est pas un arbre binaire de recherche : le fils droit de 13 vaut 4, alors qu'il devrait être supérieur à 13.

FormaV - document d'entraînement non officiel

2

Question 2. (a)

Sujet p. 1

> Recopier et compléter la ligne 3 de l'assertion de la fonction construire afin de vérifier les conditions sur les paramètres mini et maxi, qui sont des entiers respectant mini ≤ maxi. La fonction isinstance(obj, t) renvoie True si obj est du type t, sinon False.

Il faut vérifier que `mini` et `maxi` sont chacun de type `int`, puis que l'ordre annoncé dans la docstring est respecté, c'est-à-dire `mini <= maxi`.

RÉPONSE À RETENIR

```
assert isinstance(mini, int) and isinstance(maxi, int) and mini <= maxi
```

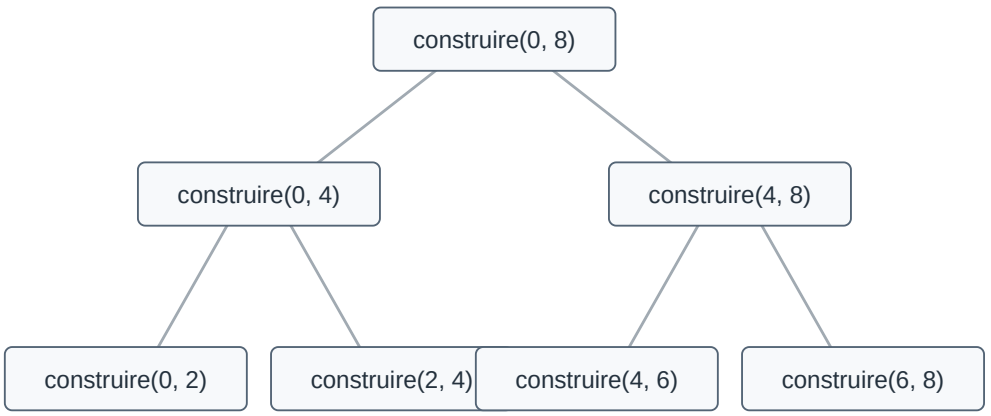
Question 2. (b)

Sujet p. 1

> On exécute l'instruction construire(0, 8). Déterminer les différents appels récursifs de la fonction construire exécutés. Représenter ces appels sous forme d'une arborescence.

Pour un intervalle de largeur strictement supérieure à 2, la fonction calcule le milieu entier `(mini + maxi) // 2`, puis appelle `construire` sur les deux intervalles déterminés par ce milieu. Ainsi, l'appel initial `construire(0, 8)` produit les appels `construire(0, 4)` et `construire(4, 8)`. Chacun produit deux appels terminaux sur des intervalles de largeur 2.

Appels de construire(0, 8)



Chaque appel sur un intervalle de largeur 2 est terminal : il ne déclenche pas d'autre appel récursif.

RÉPONSE À RETENIR

Les appels exécutés, y compris les appels terminaux, sont représentés ci-dessous :

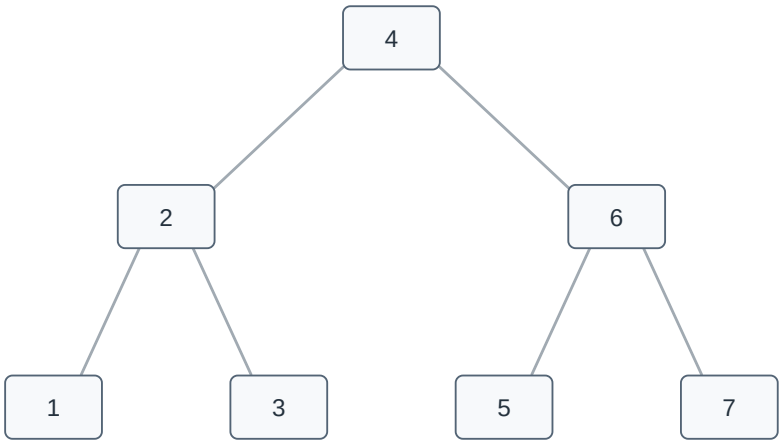
Question 2. (c)

Sujet p. 1, 2

> Dessiner l'arbre renvoyé par l'instruction construire(0, 8).

L'appel initial place à la racine le milieu `(0 + 8) // 2 = 4`. Les appels sur les intervalles `(0, 4)` et `(4, 8)` placent respectivement les valeurs 2 et 6. Les appels terminaux placent les valeurs 1, 3, 5 et 7.

Arbre renvoyé par construire(0, 8)



Les arêtes indiquent les liens entre chaque nœud et ses fils gauche et droit.

> **RÉPONSE À RETENIR**
>
> L'arbre renvoyé est le suivant :

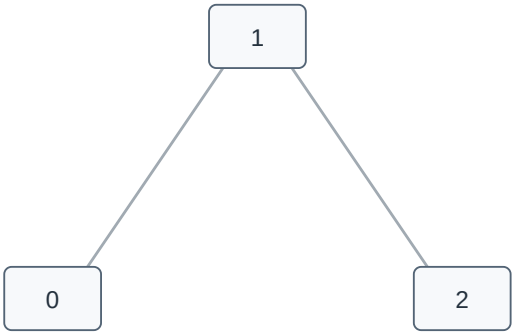
Question 2. (d)

Sujet p. 1, 2

> Dessiner l'arbre renvoyé par l'instruction construire(0, 3).

La différence initiale vaut 3, donc la fonction prend le cas général et place à la racine le milieu entier `(0 + 3) // 2 = 1`. L'appel sur `(0, 1)` renvoie un nœud de valeur 0, tandis que l'appel sur `(1, 3)`, de différence 2, renvoie un nœud de valeur 2.

Arbre renvoyé par construire(0, 3)



Le nœud 0 est le fils gauche de 1 et le nœud 2 son fils droit.

RÉPONSE À RETENIR

L'arbre renvoyé est le suivant :

Question 2. (e)

Sujet p. 1, 2

> Donner le résultat d'un parcours infixe sur l'arbre obtenu à la question 2.c. Expliquer pourquoi ce parcours permet d'affirmer que l'arbre est un arbre binaire de recherche.

Un parcours infixe visite le sous-arbre gauche, puis le nœud, puis le sous-arbre droit. Sur l'arbre de la question 2.c, il visite donc les valeurs 1, 2, 3, puis 4, puis 5, 6, 7. Pour un arbre dont les valeurs sont distinctes, le parcours infixe est strictement croissant si et seulement si l'arbre respecte la propriété d'ABR : les valeurs des sous-arbres gauches sont inférieures au nœud et celles des sous-arbres droits lui sont supérieures.

RÉPONSE À RETENIR

Le parcours infixe donne `1, 2, 3, 4, 5, 6, 7`. Il est strictement croissant ; l'arbre obtenu est donc un arbre binaire de recherche.

Question 2. (f)

Sujet p. 2

> La fonction récursive maximum prend en paramètre un arbre binaire de recherche abr et renvoie la valeur maximale de ses nœuds. Recopier et compléter les lignes 5 et 7 de cette fonction.

Dans un ABR, la valeur maximale se trouve en suivant les fils droits jusqu'au dernier nœud. Si le nœud courant n'a pas de fils droit, sa propre valeur est le maximum ; sinon, il faut chercher le maximum dans son sous-arbre droit.

RÉPONSE À RETENIR

```
5       return abr.valeur
7       return maximum(abr.droit)
```

La ligne 5 est `return abr.valeur` et la ligne 7 est `return maximum(abr.droit)`.

3

Question 3. (a)

Sujet p. 2

Pour l'arbre binaire de recherche abr_7_noeuds représenté sur la page, donner les résultats obtenus lorsque l'on exécute les instructions mystere(abr_7_noeuds, 5, []), puis mystere(abr_7_noeuds, 6, []), puis mystere(abr_7_noeuds, 2, []).

La fonction ajoute à `liste` la valeur de chaque nœud visité, puis suit la branche déterminée par la comparaison avec `x`. Si elle trouve `x`, elle renvoie la liste du chemin depuis la racine. Si elle atteint `None`, elle renvoie une nouvelle liste vide.

- Pour `x = 5`, le chemin visité est 6, 4, 5.
- Pour `x = 6`, la valeur est trouvée à la racine.
- Pour `x = 2`, la fonction visite 6, 4, 3, puis atteint un fils absent et renvoie `[]`.

RÉPONSE À RETENIR

Les résultats, dans l'ordre demandé, sont :

- `mystere(abr_7_noeuds, 5, [])` renvoie `[6, 4, 5]` ;
- `mystere(abr_7_noeuds, 6, [])` renvoie `[6]` ;
- `mystere(abr_7_noeuds, 2, [])` renvoie `[]`.

Dans le dernier cas, la liste initialement passée en argument a néanmoins été modifiée pendant les appels et contient `[6, 4, 3]` ; la valeur renvoyée est bien une autre liste vide.

Question 3. (b)

Sujet p. 2

Décrire quel peut être le rôle de la fonction mystere donnée sur la page.

La fonction compare `x` à chaque nœud et, grâce à la propriété d'ABR, ne poursuit que dans le sous-arbre où `x` peut se trouver. Si elle trouve cette valeur, elle renvoie les valeurs du chemin parcouru depuis la racine jusqu'au nœud recherché. Si la valeur est absente, elle finit par atteindre `None` et renvoie une liste vide.

RÉPONSE À RETENIR

`mystere` sert à rechercher `x` dans l'ABR. Si `x` est présent, la fonction renvoie le chemin des valeurs depuis la racine jusqu'à lui ; si `x` est absent, elle renvoie `[]`.

FormaV - document d'entraînement non officiel

Exercice 2

4 points

Bases de données et SQL

1

Question 1. (a)

Sujet p. 3, 4

> Dans cette partie, on ne demande pas de requête SQL. En étudiant le contenu des tables, indiquer quels sont les prénoms et noms du ou des membres du site qui proposent la location d'un appareil à raclette.

Dans la table `Objet`, l'appareil à raclette a l'identifiant 4. On cherche alors, dans `Possede`, les lignes dont `id_objet` vaut 4 : elles indiquent les membres 1 et 2. La table `Membre` permet de retrouver leurs noms et prénoms.

RÉPONSE À RETENIR

Les membres sont Mohamed Ali et Fernando Alonso.

Question 1. (b)

Sujet p. 3, 4

> Dans cette partie, on ne demande pas de requête SQL. En étudiant le contenu des tables, donner le prénom et le nom du membre qui ne propose pas d'objet à la location.

Les membres qui proposent au moins un objet apparaissent dans la colonne `id_membre` de la table `Possede` : ce sont les membres 1, 2, 3 et 4. Le membre 5 n'y apparaît pas ; il ne propose donc aucun objet.

RÉPONSE À RETENIR

Il s'agit de Harry Kane.

2

Question 2. (a)

Sujet p. 3, 4

> Donner le résultat de la requête suivante : SELECT nom, prenom FROM Membre WHERE cp = "69003" ;

La requête sélectionne les noms et prénoms des membres dont le code postal est `69003`. Dans la table `Membre`, cela correspond aux membres Antoine Dupont et Harry Kane. Les colonnes sont affichées dans l'ordre demandé par la requête : `nom`, puis `prenom`.

RÉPONSE À RETENIR

nom	prenom
Dupont	Antoine
Kane	Harry

FormaV - document d'entraînement non officiel

Question 2. (b)

Sujet p. 3, 4

> Écrire une requête permettant de connaître le tarif de location d'une scie circulaire.

La description de la scie circulaire se trouve dans la table `Objet`. Il faut donc sélectionner son tarif en filtrant sur cette description. Dans les données fournies, la scie circulaire a un tarif de 15.

```
SELECT tarif
FROM Objet
WHERE description = 'Scie circulaire';
```

RÉPONSE À RETENIR

Résultat sur les données du sujet :

tarif
15

Question 2. (c)

Sujet p. 3, 4

> Écrire une requête permettant de modifier le tarif de location d'un nettoyeur à haute pression pour le passer à 15 € par jour au lieu de 20 € par jour.

La modification se fait avec `UPDATE` et `SET`. Le filtre sur la description cible le nettoyeur haute pression sans modifier le tarif des autres objets.

```
UPDATE Objet
SET tarif = 15
WHERE description = 'Nettoyeur haute pression';
```

RÉPONSE À RETENIR

Le tarif journalier du nettoyeur haute pression est ainsi fixé à 15 €.

Question 2. (d)

Sujet p. 3, 4

> Écrire une requête SQL permettant d'ajouter Wendie Renard habitant à Villeurbanne (code postal 69100) dans la table Membre, avec un id_membre de 6.

On insère une ligne dans `Membre` en indiquant les colonnes et leurs valeurs dans le même ordre : identifiant, nom, prénom et code postal.

```
INSERT INTO Membre (id_membre, nom, prenom, cp)
VALUES (6, 'Renard', 'Wendie', '69100');
```

RÉPONSE À RETENIR

La table contient alors le membre d'identifiant 6, Wendie Renard, dont le code postal est 69100.

3

Question 3. (a)

Sujet p. 3, 4

> Expliquer la limitation importante d'utilisation du service offert par le site si l'on utilisait le couple de clés étrangères (id_objet, id_membre) en tant que clé primaire de la relation Reservation.

Une clé primaire doit être unique. Si le couple `(id_objet, id_membre)` était la clé primaire de `Reservation`, un même membre ne pourrait apparaître qu'une seule fois pour un même objet dans cette table. Il lui serait donc impossible de réserver à nouveau ce même objet à une autre date. Les réservations 5 et 6 du tableau illustrent le cas : elles concernent le même objet 2 et le même membre 2, mais sont deux réservations distinctes.

> **RÉPONSE À RETENIR**
>
> Cette clé primaire empêcherait donc les réservations répétées d'un même objet par un même membre, même à des dates différentes.

Question 3. (b)

Sujet p. 3, 4

> Mohamed Ali décide de ne plus être membre du site. Il faut donc le supprimer de la table Membre à l'aide de la requête DELETE FROM Membre WHERE nom = "Ali" AND prenom = "Mohamed" ;. Expliquer pourquoi cette requête produit une erreur.

La ligne de Mohamed Ali dans `Membre` est référencée par des lignes d'autres tables : l'identifiant 1 apparaît dans `Possede` et dans `Reservation`. Ces colonnes sont des clés étrangères qui doivent référencer un membre existant. Sans suppression préalable des lignes qui les utilisent (ou mécanisme explicite de suppression en cascade), supprimer directement ce membre violerait l'intégrité référentielle.

> **RÉPONSE À RETENIR**
>
> La requête échoue parce que des clés étrangères dans `Possede` et `Reservation` référencent encore le membre d'identifiant 1.

Question 3. (c)

Sujet p. 3, 4

> Proposer une suite de requêtes utilisant le mot-clé DELETE, précédant la requête ci-dessus, pour supprimer correctement Mohamed Ali, dont l'id_membre est 1, de la base de données. La relation Objet décrit le type d'objet à la location. Il n'y a pas d'objet à supprimer dans cette table lors du départ d'un membre.

Il faut supprimer d'abord les réservations et les liens de possession associés à l'identifiant 1, puis exécuter la requête de suppression du membre fournie dans l'énoncé. La table `Objet` n'est pas supprimée : elle décrit les types d'objets, qui peuvent rester proposés par d'autres membres.

```
DELETE FROM Reservation
WHERE id_membre = 1;

DELETE FROM Possede
WHERE id_membre = 1;

DELETE FROM Membre
WHERE nom = 'Ali' AND prenom = 'Mohamed';
```


FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Les références au membre étant supprimées avant sa ligne dans `Membre`, la suppression de Mohamed Ali peut respecter les contraintes de clés étrangères.

4

Question 4. (a)

Sujet p. 3, 4

> Dans cette partie, les requêtes utilisent des jointures entre tables. On supposera donc les numéros id_membre et id_objet non connus. Écrire une requête permettant de compter le nombre de réservations réalisées par Fernando Alonso.

On joint `Membre` à `Reservation` grâce à leur colonne commune `id_membre`, puis on filtre sur le nom et le prénom de Fernando Alonso. Enfin, `COUNT(*)` compte ses réservations. Dans les données fournies, son identifiant est 2 et trois lignes de `Reservation` portent cet identifiant.

```
SELECT COUNT(*) AS nombre_reservations
FROM Membre
JOIN Reservation ON Membre.id_membre = Reservation.id_membre
WHERE Membre.nom = 'Alonso'
  AND Membre.prenom = 'Fernando';
```

RÉPONSE À RETENIR

Résultat sur les données du sujet :

nombre_reservations
3

Question 4. (b)

Sujet p. 3, 4

> Dans cette partie, les requêtes utilisent des jointures entre tables. On supposera donc les numéros id_membre et id_objet non connus. Écrire une requête permettant de connaître les noms et prénoms des membres possédant un appareil à raclette.

La table `Possede` relie les membres aux objets. On la joint à `Membre` par `id_membre`, puis à `Objet` par `id_objet`. Le filtre sur la description sélectionne l'appareil à raclette. `DISTINCT` évite de répéter un membre si plusieurs lignes correspondaient.

```
SELECT DISTINCT Membre.nom, Membre.prenom
FROM Membre
JOIN Possede ON Membre.id_membre = Possede.id_membre
JOIN Objet ON Possede.id_objet = Objet.id_objet
WHERE Objet.description = 'Appareil à raclette';
```

RÉPONSE À RETENIR

Résultat sur les données du sujet :

nom	prenom
Ali	Mohamed
Alonso	Fernando

FormaV - document d'entraînement non officiel

Exercice 3

4 points

Architecture matérielle, processus et ordonnancement

1

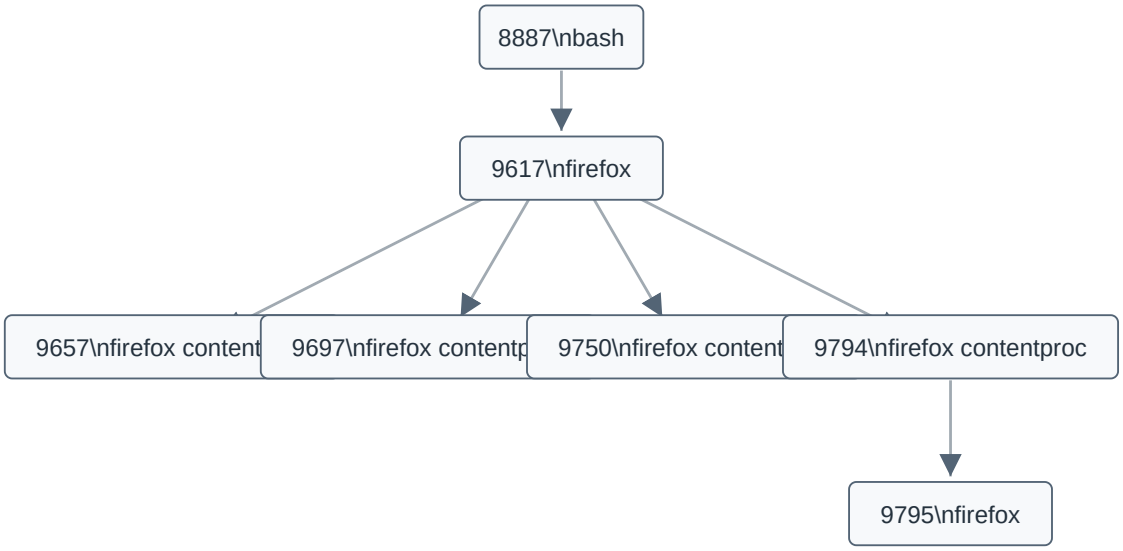
Question 1. (a)

Sujet p. 5

> Avec la commande ps -aef, on obtient l'affichage fourni. Donner sous forme d'un arbre de PID la hiérarchie des processus liés à firefox. PID désigne l'identifiant d'un processus et PPID l'identifiant de son processus parent.

Dans le tableau, le champ PPID indique le PID du processus parent. Ainsi, 9617 a pour parent 8887 ; 9657, 9697, 9750 et 9794 ont pour parent 9617 ; enfin, 9795 a pour parent 9794. Le processus 8887 est le shell `bash`.

Hiérarchie des processus liés à Firefox



Les PID et les liens parent-enfant sont ceux indiqués dans l'affichage de ps -aef.

RÉPONSE À RETENIR

La hiérarchie des processus liés à Firefox est représentée ci-dessous. Les flèches vont du parent vers l'enfant.

Question 1. (b)

Sujet p. 5

> À partir de l'affichage de la commande ps -aef, indiquer la commande qui a lancé le premier processus de firefox.

Le premier processus Firefox de la hiérarchie est celui de PID 9617. La colonne CMD de sa ligne donne la commande qui l'a lancé.

RÉPONSE À RETENIR

`/usr/lib/firefox/firefox`

Question 1. (c)

Sujet p. 5

> La commande kill permet de supprimer un processus à l'aide de son PID. Indiquer la commande qui permettra de supprimer tous les processus liés à firefox et uniquement cela.

Il faut viser les six processus Firefox indiqués, sans inclure leur parent `bash` de PID 8887. On peut donner les PID des processus descendants avant celui de leur parent Firefox.

RÉPONSE À RETENIR

Une commande possible est :

```
kill 9795 9657 9697 9750 9794 9617
```

Elle cible uniquement les PID des processus liés à Firefox présents dans le tableau. `kill` envoie un signal de terminaison ; la disparition effective des processus peut ne pas être instantanée.

2

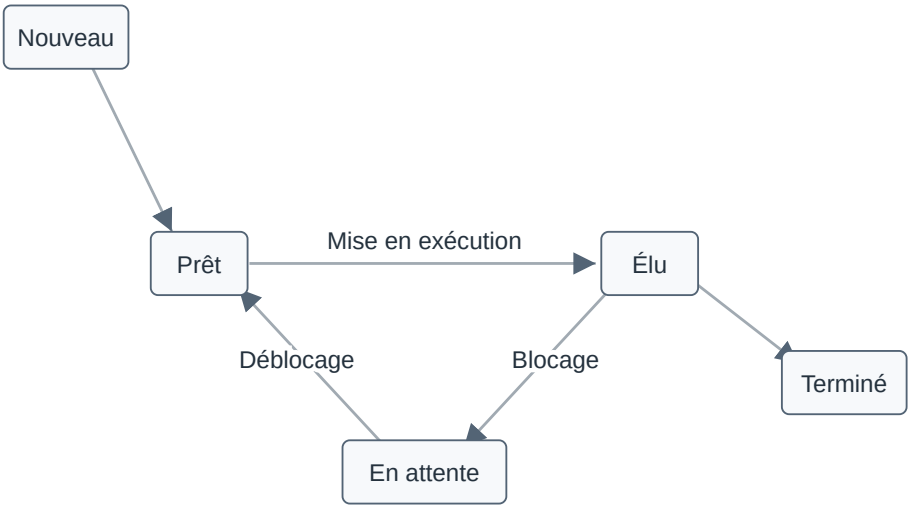
Question 2. (a)

Sujet p. 5

> Recopier et compléter le schéma d'ordonnancement des processus avec les termes suivants : Élu, En attente, Prêt, Blocage, Déblocage, Mise en exécution.

Un processus passe de Nouveau à Prêt, puis peut être sélectionné par l'ordonnanceur et devenir Élu. Lorsqu'un processus élu doit attendre un événement, il passe en En attente ; après le déblocage, il redevient Prêt. Un processus élu peut aussi terminer. Les trois états à inscrire dans les ovales sont donc Prêt à gauche, Élu à droite et En attente en bas.

États et transitions d'un processus



Les flèches Nouveau → Prêt et Élu → Terminé sont déjà représentées dans le sujet ; les autres portent les transitions à placer.

RÉPONSE À RETENIR

Les transitions à compléter sont : Prêt vers Élu : « Mise en exécution » ; Élu vers En attente : « Blocage » ; En attente vers Prêt : « Déblocage ». Le schéma indique également Nouveau vers Prêt et Élu vers Terminé.

Question 2. (b)

Sujet p. 5, 6

Quatre processus doivent être exécutés par un processeur. Leurs instants d'arrivée sont P1 : 0, P2 : 2, P3 : 3 et P4 : 7 ; leurs durées sont respectivement 8, 6, 2 et 2 cycles. L'ordonnanceur choisit parmi les processus présents en liste d'attente celui dont la durée restante est la plus courte, exécute un cycle, puis désigne de nouveau le processus dont la durée restante est la plus courte. En cas d'égalité, il choisit celui dont l'instant d'arrivée est le plus ancien. À partir de l'ordonnancement fourni, calculer la moyenne des temps d'exécution des quatre processus. Le temps d'exécution d'un processus est la différence entre son instant de terminaison et son instant d'arrivée.

On lit le diagramme fourni cycle par cycle : P1 s'exécute de 0 à 3, P3 de 3 à 5, P1 de 5 à 7, P4 de 7 à 9, P1 de 9 à 12 et P2 de 12 à 18. Les dates de terminaison sont donc P1 : 12, P2 : 18, P3 : 5 et P4 : 9. On calcule chaque temps d'exécution en soustrayant l'instant d'arrivée à l'instant de terminaison.

Processus	Terminaison	Arrivée	Temps d'exécution
P1	12	0	12 − 0 = 12
P2	18	2	18 − 2 = 16
P3	5	3	5 − 3 = 2
P4	9	7	9 − 7 = 2

La somme des quatre temps est 12 + 16 + 2 + 2 = 32 cycles ; on divise cette somme par 4 pour obtenir la moyenne.

RÉPONSE À RETENIR

La moyenne des temps d'exécution est $32/4 = 8$ cycles.

Question 2. (c)

Sujet p. 5, 6

On modifie l'ordonnancement : l'algorithme reste identique, mais le processeur exécute à chaque fois deux cycles du processus choisi au lieu d'un. En cas d'égalité du temps restant, l'ordonnanceur départage toujours en fonction de l'instant d'arrivée. Recopier et compléter le schéma donnant le nouvel ordonnancement des quatre processus.

On simule le choix à chaque tranche de deux cycles, en tenant compte des arrivées et des durées restantes. De 0 à 2, P1 s'exécute. À 2, P1 et P2 ont chacun 6 cycles restants ; P1 étant arrivé le premier, il est choisi de 2 à 4. À 4, P3 a 2 cycles restants et est choisi de 4 à 6. P1, avec 4 cycles restants, est ensuite choisi de 6 à 8. À 8, P1 et P4 ont chacun 2 cycles restants ; P1, arrivé le premier, s'exécute de 8 à 10 et termine. P4 s'exécute ensuite de 10 à 12, puis P2 termine par trois tranches de deux cycles.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Le nouvel ordonnancement est :

Intervalle de temps	Processus
0 à 2	P1
2 à 4	P1
4 à 6	P3
6 à 8	P1
8 à 10	P1
10 à 12	P4
12 à 14	P2
14 à 16	P2
16 à 18	P2

Question 2. (d)

Sujet p. 5, 6

Calculer la nouvelle moyenne des temps d'exécution des quatre processus et indiquer si cet ordonnancement est plus performant que le précédent.

Dans le nouvel ordonnancement, les dates de terminaison sont P1 : 10, P2 : 18, P3 : 6 et P4 : 12. Les temps d'exécution valent donc respectivement 10 − 0 = 10, 18 − 2 = 16, 6 − 3 = 3 et 12 − 7 = 5 cycles. Leur somme est 34 cycles.

RÉPONSE À RETENIR

La nouvelle moyenne est $34/4 = 8{,}5$ cycles. Elle est supérieure à l'ancienne moyenne de 8 cycles : selon le critère du temps moyen d'exécution, ce nouvel ordonnancement est moins performant.

3

Question 3. (a)

Sujet p. 6

Chaque processus est représenté par une liste comportant autant d'éléments que de durées en nombre de cycles. Pour simuler la date de création, on ajoute en fin de liste autant de chaînes de caractères vides que la valeur de cette date. Recopier sans les commentaires et compléter la fonction choix_processus(liste_attente), qui renvoie l'indice du processus dont le temps restant d'exécution est le plus court parmi ceux présents en liste d'attente. Le code peut contenir plusieurs lignes.

La longueur de chaque liste représente le nombre de cycles restant au processus correspondant. On initialise le minimum avec la longueur de la première liste, puis on parcourt les autres listes en mettant à jour le minimum et son indice lorsqu'on trouve une liste plus courte. La comparaison stricte conserve le premier indice en cas d'égalité ; puisque les processus sont ajoutés à la liste d'attente dans leur ordre d'arrivée, cela respecte la règle de départage.

FormaV - document d'entraînement non officiel

```
def choix_processus(liste_attente):
    """Renvoie l'indice du processus le plus court parmi
    ceux présents en liste d'attente liste_attente"""
    if liste_attente != []:
        mini = len(liste_attente[0])
        indice = 0
        for i in range(1, len(liste_attente)):
            if len(liste_attente[i]) < mini:
                mini = len(liste_attente[i])
                indice = i
        return indice
```

La fonction examine une fois chaque liste en attente : pour n processus présents, elle effectue un nombre d'opérations proportionnel à n, avec une mémoire supplémentaire constante. Si la liste est vide, elle renvoie implicitement `None` ; l'ordonnancement ne l'appelle que lorsque la liste n'est pas vide.

> **RÉPONSE À RETENIR**
>
> Trace avec les données du sujet, en supposant l'ordre d'arrivée des processus dans la liste d'attente : à l'instant 0, seul P1 est présent, donc l'indice renvoyé est 0. À l'instant 2, P1 et P2 ont chacun 6 cycles restants ; l'indice 0 de P1 est conservé. À l'instant 3, P1 a 5 cycles restants, P2 en a 6 et P3 en a 2 ; l'indice renvoyé est alors 2, celui de P3 dans la liste [P1, P2, P3].

Question 3. (b)

Sujet p. 6, 7

> Recopier et compléter les différentes instructions de la fonction ordonnancement pour réaliser le fonctionnement désiré.

À chaque tour, le premier élément de la liste choisie représente le prochain cycle à exécuter. Son premier caractère identifie le processus (par exemple, `'1.8'[0]` vaut `'1'`). On ajoute cet identifiant à `execution`, puis on retire l'élément consommé. Si la liste du processus est devenue vide, on la retire de la liste d'attente avant la nouvelle scrutation.

```
def ordonnancement(liste_proc):
    """Exécute l'algorithme d'ordonnancement
    liste_proc -- liste des processus
    Renvoie la liste d'exécution des processus"""
    execution = []
    attente = scrutation(liste_proc, [])
    while attente != []:
        indice = choix_processus(attente)
        execution.append(attente[indice][0][0])
        attente[indice].pop(0)
        if attente[indice] == []:
            attente.pop(indice)
        attente = scrutation(liste_proc, attente)
    return execution
```

À chaque itération, un cycle d'un processus prêt est consommé. Les listes en attente représentent les processus arrivés et non terminés, et leurs éléments représentent leurs cycles restants. Comme le nombre total de cycles est fini et que les processus sont ajoutés à leur arrivée par `scrutation`, l'algorithme termine.

RÉPONSE À RETENIR

Avec les processus du sujet, la suite des exécutions par cycle est :

```
['1', '1', '1', '3', '3', '1', '1', '4', '4', '1', '1', '1', '2', '2', '2', '2', '2',
'2']
```

Elle correspond aux intervalles P1 de 0 à 3, P3 de 3 à 5, P1 de 5 à 7, P4 de 7 à 9, P1 de 9 à 12 et P2 de 12 à 18.

Sujet : https://aeif.fr/wp-content/uploads/2025/07/Polynesie_mars_2023_2.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.