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 2023 • Polynésie • Jour 2

Code sujet **23-NSIJ2PO1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Arbres binaires, arbres binaires de recherche, programmation orientée objet et récursivité
10 réponses • sujet p. 2–4

Exercice 2 — Base de données : modèle relationnel et langage SQL
11 réponses • sujet p. 5–7

Exercice 3 — Architecture matérielle : processus et ordonnancement
9 réponses • sujet p. 8–10

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, arbres binaires de recherche, programmation orientée objet et récursivité

1

Question a.

Sujet p. 2

> Justifier que l'arbre représenté est un arbre binaire.

Un arbre binaire est un arbre dans lequel chaque nœud possède au plus deux fils, appelés fils gauche et fils droit. Sur l'arbre représenté, chaque nœud a au plus deux fils.

RÉPONSE À RETENIR

L'arbre représenté est donc bien un arbre binaire.

Question b.

Sujet p. 2

> Indiquer si l'arbre représenté est un arbre binaire de recherche (ABR) et justifier la réponse.

Dans un arbre binaire de recherche, toutes les valeurs du sous-arbre droit d'un nœud doivent être supérieures à la valeur de ce nœud. Or la racine vaut 13 et son fils droit vaut 4, avec $4 < 13$: la propriété est déjà violée.

RÉPONSE À RETENIR

Cet arbre n'est pas un arbre binaire de recherche (ABR).

2

Question a.

Sujet p. 2, 3

> Recopier et compléter la ligne 3 de l'assertion de la fonction construire afin de vérifier les conditions sur les paramètres mini et maxi.

La ligne doit vérifier que `mini` et `maxi` sont des entiers et que l'inégalité indiquée dans la docstring, `mini <= maxi`, est respectée. Les conditions sont reliées par `and`.

RÉPONSE À RETENIR

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

Question b.

Sujet p. 2, 3

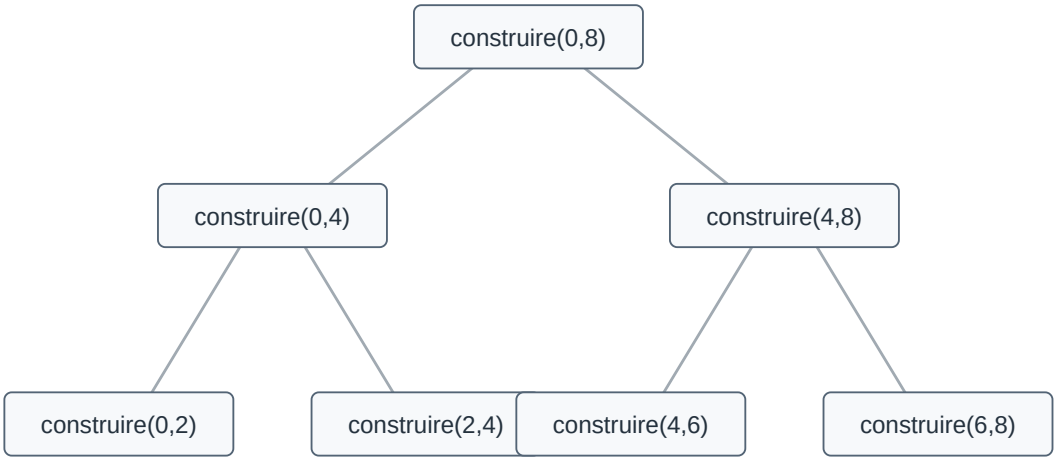
> On exécute l'instruction construire(0,8). Déterminer les différents appels récursifs de la fonction construire exécutés et les représenter sous forme d'une arborescence.

Pour un appel dont l'écart entre les paramètres est supérieur à 2, la fonction calcule le milieu entier, puis lance un appel sur l'intervalle de gauche et un autre sur l'intervalle de droite. Ainsi, l'appel initial `construire(0,8)` a pour milieu 4 et donne les appels `construire(0,4)` et `construire(4,8)`. Chacun de

FormaV - document d'entraînement non officiel

ces appels a ensuite un écart de 4, puis produit deux appels d'écart 2. Ces quatre derniers appels s'arrêtent au cas `maxi - mini == 2`.

Arborescence des appels de construire(0,8)



Chaque nœud représente un appel de la fonction.

> **RÉPONSE À RETENIR**
>
> Les appels récursifs sont ceux représentés dans l'arborescence ci-dessous ; les appels terminaux sont `construire(0,2)`, `construire(2,4)`, `construire(4,6)` et `construire(6,8)`.

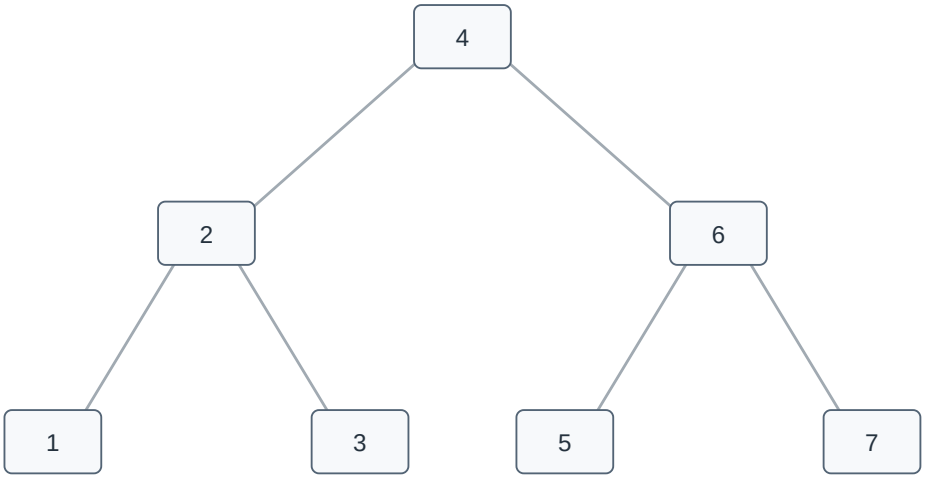
Question c.

Sujet p. 2, 3

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

L'appel `construire(0,8)` crée un nœud de valeur `(0+8)//2 = 4`. Son fils gauche est construit par `construire(0,4)`, qui a pour valeur centrale 2 ; son fils droit est construit par `construire(4,8)`, qui a pour valeur centrale 6. Enfin, les appels terminaux donnent respectivement les valeurs 1, 3, 5 et 7.

Arbre renvoyé par construire(0,8)



Les fils gauche et droit correspondent aux sous-intervalles utilisés par la fonction.

> **RÉPONSE À RETENIR**
>
> L'arbre renvoyé a pour racine 4, pour fils gauche 2 et pour fils droit 6. Les fils de 2 sont 1 et 3 ; ceux de 6 sont 5 et 7.

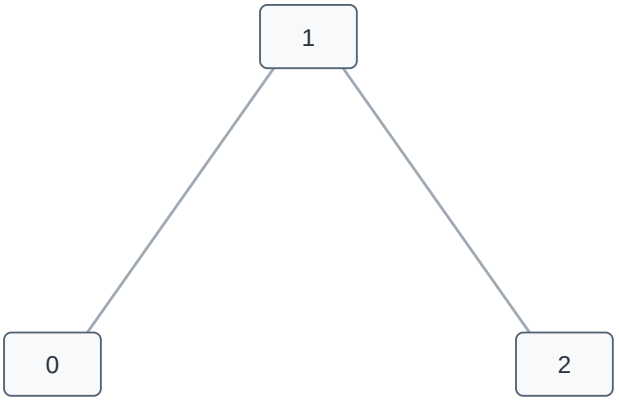
Question d.

Sujet p. 2, 3

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

Pour `construire(0,3)`, l'écart vaut 3. La fonction calcule le milieu entier `(0+3)//2 = 1`, puis appelle `construire(0,1)` et `construire(1,3)`. Ces appels renvoient des nœuds de valeurs 0 et 2, car leurs écarts valent respectivement 1 et 2.

Arbre renvoyé par construire(0,3)



Le fils gauche vaut 0 et le fils droit vaut 2.

> **RÉPONSE À RETENIR**
>
> L'arbre renvoyé a pour racine 1, pour fils gauche 0 et pour fils droit 2.

Question e.

Sujet p. 2, 3

> 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 successivement le sous-arbre gauche, le nœud, puis le sous-arbre droit. Sur l'arbre de la question 2.c, cela donne d'abord 1, 2, 3, puis 4, puis 5, 6, 7. Pour des valeurs distinctes, un parcours infixe strictement croissant signifie que, pour chaque nœud, les valeurs de son sous-arbre gauche sont plus petites que la sienne et celles de son sous-arbre droit sont plus grandes : c'est la propriété d'un ABR.

> **RÉPONSE À RETENIR**
>
> Le parcours infixe donne `1, 2, 3, 4, 5, 6, 7`. La suite est strictement croissante ; l'arbre obtenu est donc un ABR.

Question f.

Sujet p. 3

> 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, le maximum se trouve en suivant les fils droits jusqu'au dernier nœud. Si le nœud courant n'a pas de fils droit, sa valeur est donc le maximum. Sinon, on poursuit récursivement la recherche dans son sous-arbre droit.

RÉPONSE À RETENIR

Les lignes complétées sont :

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

3

Question a.

Sujet p. 4

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

L'arbre a pour racine 6. Son fils gauche est 4, dont les fils sont 3 et 5 ; son fils droit est 8, dont les fils sont 7 et 9. La fonction ajoute à `liste` chaque valeur rencontrée, puis suit le fils gauche ou droit selon la comparaison avec `x`. Si elle atteint un arbre vide, elle renvoie une nouvelle liste vide.

Pour `x = 5`, les nœuds visités sont 6, 4 et 5. Pour `x = 6`, la valeur est trouvée à la racine. Pour `x = 2`, les nœuds 6, 4 et 3 sont ajoutés à la liste, puis la recherche atteint un fils vide.

RÉPONSE À RETENIR

Dans l'ordre demandé :

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

Dans le troisième appel, la liste fournie est toutefois modifiée en place : elle contient `[6, 4, 3]` à la fin, même si la valeur renvoyée est la nouvelle liste vide.

Question b.

Sujet p. 4

Décrire quel peut être le rôle de la fonction mystere.

La fonction suit dans l'ABR le chemin de recherche de la valeur `x`. Si elle trouve cette valeur, elle renvoie la liste des valeurs rencontrées depuis la racine jusqu'au nœud trouvé. Si elle ne la trouve pas, elle renvoie une liste vide. Comme la liste passée en argument est modifiée en place, elle peut néanmoins contenir les valeurs du chemin parcouru après un échec.

RÉPONSE À RETENIR

`mystere` peut servir à rechercher une valeur dans un ABR et, si elle y figure, à renvoyer le chemin suivi depuis la racine jusqu'à cette valeur.

Exercice 2

4 points

Base de données : modèle relationnel et langage SQL

1

Question a.

Sujet p. 5, 6

> Dans cette partie, aucune requête SQL n'est demandée. En étudiant le contenu des tables fournies, indiquer quels sont les prénoms et noms du ou des membres du site qui proposent la location d'un appareil à raclette.

Dans `Objet`, l'appareil à raclette a l'identifiant 4. Dans `Possede`, cet objet est associé aux membres d'identifiants 1 et 2. En retrouvant ces identifiants dans `Membre`, on obtient leurs prénoms et leurs noms.

RÉPONSE À RETENIR

Les membres qui proposent un appareil à raclette sont Mohamed Ali et Fernando Alonso.

Question b.

Sujet p. 5, 6

> Dans cette partie, aucune requête SQL n'est demandée. En étudiant le contenu des tables fournies, donner le prénom et le nom du membre qui ne propose pas d'objet à la location.

Les membres qui proposent des objets apparaissent dans la colonne `id_membre` de `Possede`. L'identifiant 5, qui correspond à Harry Kane dans `Membre`, n'y apparaît pas. Le fait qu'il ait effectué une réservation ne signifie pas qu'il propose un objet.

RÉPONSE À RETENIR

Harry Kane ne propose pas d'objet à la location.

2

Question a.

Sujet p. 6, 7

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

La requête conserve les membres dont le code postal est `69003`, puis affiche leurs colonnes `nom` et `prenom`. Les lignes correspondantes sont celles d'Antoine Dupont et de Harry Kane.

RÉPONSE À RETENIR

nom	prenom
Dupont	Antoine
Kane	Harry

FormaV - document d'entraînement non officiel

Question b.

Sujet p. 5, 6, 7

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

On filtre la table `Objet` sur la description de la scie circulaire et on sélectionne la colonne `tarif`.

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

RÉPONSE À RETENIR

Dans les données fournies, le résultat est le tarif journalier de 15 €.

tarif
15

Question c.

Sujet p. 5, 6, 7

> 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.

On met à jour la colonne `tarif` dans la ligne de l'objet identifié par sa description. La description figurant dans la table est « Nettoyeur haute pression ».

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

RÉPONSE À RETENIR

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

Question d.

Sujet p. 5, 6, 7

> 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 explicitement l'ordre des colonnes : identifiant, nom, prénom et code postal. Le code postal est écrit comme une chaîne de caractères, conformément au type `TEXT` de la colonne `cp`.

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

RÉPONSE À RETENIR

La ligne ajoutée correspond à Wendie Renard, de code postal 69100 et d'identifiant 6.

3

Question a.

Sujet p. 5, 7

> 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 impose que chaque valeur de clé soit unique. Si `(id_objet, id_membre)` est la clé primaire de `Reservation`, le même membre ne peut pas avoir deux réservations du même objet : ces deux réservations auraient le même couple d'identifiants, même si leurs dates diffèrent.

RÉPONSE À RETENIR

La limitation est qu'un membre ne pourrait réserver un même objet qu'une seule fois au total, y compris à des dates différentes.

Question b.

Sujet p. 5, 7

> 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.

L'identifiant 1 de Mohamed Ali est encore référencé comme clé étrangère dans `Possede` et dans `Reservation`. Supprimer sa ligne de `Membre` avant les lignes qui la référencent violerait l'intégrité référentielle et provoque une erreur en l'absence de suppression en cascade.

RÉPONSE À RETENIR

La suppression échoue car des lignes de `Possede` et de `Reservation` référencent encore le membre d'identifiant 1.

Question c.

Sujet p. 5, 7

> 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 d'abord supprimer les réservations puis les associations de possession qui référencent le membre 1. On peut ensuite supprimer sa ligne dans `Membre`. Il ne faut pas supprimer d'objet dans `Objet`.

```
DELETE FROM Reservation
WHERE id_membre = 1;

DELETE FROM Possede
WHERE id_membre = 1;

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

RÉPONSE À RETENIR

Cette séquence supprime les lignes dépendantes avant la ligne de Mohamed Ali dans `Membre`, tout en conservant les objets de la table `Objet`.

FormaV - document d'entraînement non officiel

4

Question a.

Sujet p. 5, 6, 7

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

La clé primaire `Membre.id_membre` est référencée par la clé étrangère `Reservation.id_membre`. On joint donc ces tables sur cet attribut, on filtre sur le nom et le prénom de Fernando Alonso, puis on compte les réservations correspondantes.

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

Dans les données fournies, les réservations 2, 5 et 6 ont été réalisées par Fernando Alonso.

RÉPONSE À RETENIR

COUNT(*)
3

Question b.

Sujet p. 5, 6, 7

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

On joint `Membre` à `Possede` par `id_membre`, puis `Possede` à `Objet` par `id_objet`. On filtre sur la description de l'appareil à raclette et on sélectionne les nom et prénom des membres associés à cet objet.

```
SELECT 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

nom	prenom
Ali	Mohamed
Alonso	Fernando

FormaV - document d'entraînement non officiel

Exercice 3

4 points

Architecture matérielle : processus et ordonnancement

1

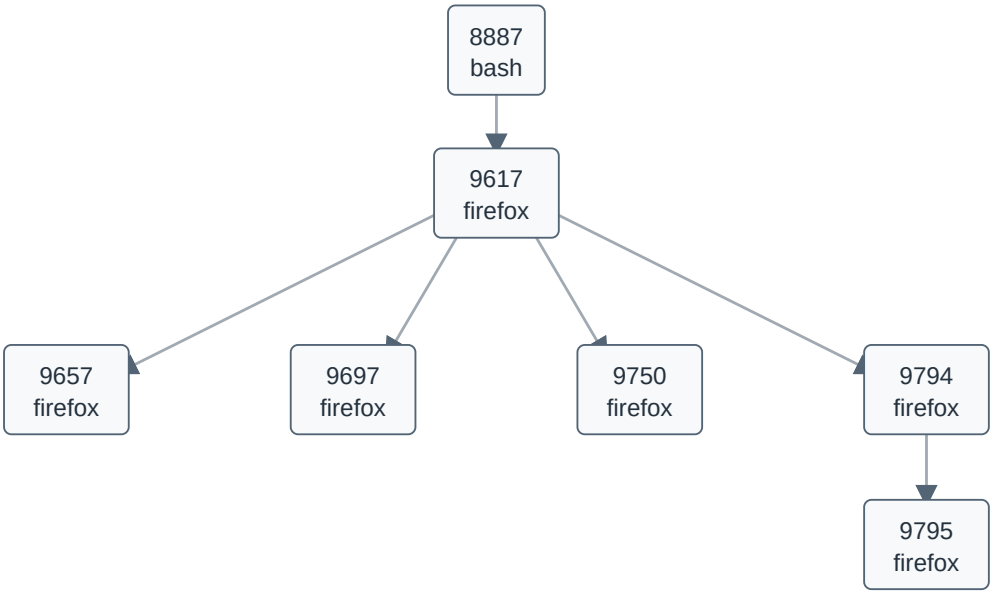
Question a.

Sujet p. 8

> À partir de l'affichage obtenu avec la commande ps -aef et des identifiants PID et PPID, donner sous forme d'un arbre de PID la hiérarchie des processus liés à firefox.

Dans l'affichage de `ps -aef`, la première colonne donne le PID et la deuxième le PPID, c'est-à-dire le PID du processus parent. Le processus Firefox de PID 9617 a pour parent le processus `bash` de PID 8887. Les processus de PID 9657, 9697, 9750 et 9794 ont pour parent 9617 ; le processus de PID 9795 a pour parent 9794.

Hiérarchie des processus liés à Firefox



Chaque flèche va du processus parent au processus enfant.

RÉPONSE À RETENIR

La hiérarchie des processus Firefox est représentée ci-dessous. Le processus `bash` de PID 8887 est indiqué comme parent du premier processus Firefox.

Question b.

Sujet p. 8

> Indiquer la commande qui a lancé le premier processus de firefox.

La ligne du PID 9617 affiche dans la colonne `CMD` le chemin de l'exécutable Firefox. Son PPID est 8887, dont la ligne correspond à `bash` : c'est donc ce shell qui a lancé le premier processus Firefox.

RÉPONSE À RETENIR

La commande est `/usr/lib/firefox/firefox`, lancée par le processus `bash` de PID 8887.

Question c.

Sujet p. 8

La commande kill permet de supprimer un processus à l'aide de son PID (par exemple kill 8600). Indiquer la commande qui permettra de supprimer tous les processus liés à firefox et uniquement cela.

Les processus à supprimer sont le processus Firefox de PID 9617 et ses descendants : 9657, 9697, 9750, 9794 et 9795. On indique explicitement ces PID pour ne viser aucun autre processus. Il est prudent d'envoyer le signal aux descendants avant leur parent.

RÉPONSE À RETENIR

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

2

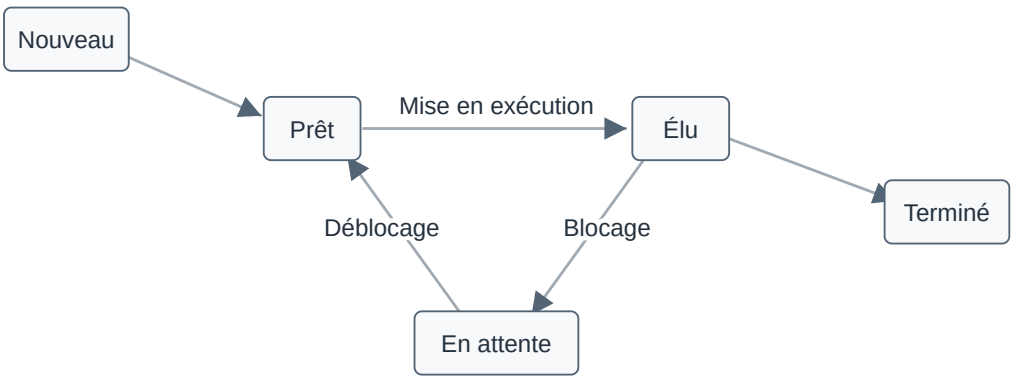
Question a.

Sujet p. 8

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

Le processus nouvellement créé passe à l'état Prêt. Lorsqu'il est sélectionné par le processeur, il passe de Prêt à Élu : cette transition est la Mise en exécution. Un processus Élu peut se terminer ou être bloqué ; dans ce dernier cas, il passe à l'état En attente. Après le Déblocage, il revient à l'état Prêt.

États et transitions d'un processus



Les trois états à compléter sont Prêt, Élu et En attente ; les transitions indiquées sont Mise en exécution, Blocage et Déblocage.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Les états et transitions à inscrire sont représentés ci-dessous.

Question b.

Sujet p. 9

Pour les quatre processus du tableau (P1, P2, P3 et P4 ; instants d'arrivée 0, 2, 3 et 7 ; durées 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, puis le processeur exécute un cycle avant une nouvelle sélection. En cas d'égalité, le processus dont l'instant d'arrivée est le plus ancien est choisi. Le temps d'exécution d'un processus est la différence entre son instant de terminaison et son instant d'arrivée. Calculer la moyenne des temps d'exécution des quatre processus.

À chaque cycle, on choisit parmi les processus déjà arrivés celui qui a le moins de cycles restants. À l'instant 2, P1 et P2 ont chacun 6 cycles restants ; l'égalité est départagée en faveur de P1, arrivé le premier. À l'instant 3, P3 arrive avec 2 cycles restants et devient prioritaire. Le choix est réévalué après chaque cycle, notamment à l'arrivée de P4 à l'instant 7.

Intervalle de temps	Processus exécuté
0 à 3	P1
3 à 5	P3
5 à 7	P1
7 à 9	P4
9 à 12	P1
12 à 18	P2

Les instants de terminaison sont respectivement 12, 18, 5 et 9 pour P1, P2, P3 et P4. Les temps d'exécution, obtenus en retranchant l'instant d'arrivée, sont donc : P1, 12 − 0 = 12 ; P2, 18 − 2 = 16 ; P3, 5 − 3 = 2 ; P4, 9 − 7 = 2 cycles.

Ordonnancement avec sélection après chaque cycle

Intervalle de temps	Processus
0–3	P1
3–5	P3
5–7	P1
7–9	P4
9–12	P1
12–18	P2

Les bornes des intervalles sont les instants de début et de fin.

RÉPONSE À RETENIR

La moyenne des temps d'exécution vaut $\frac{12+16+2+2}{4} = 8$ cycles.

FormaV - document d'entraînement non officiel

Question c.

Sujet p. 9

> L'algorithme reste identique à celui présenté précédemment, mais le processeur exécute à chaque fois deux cycles du processus choisi au lieu d'un seul. En cas d'égalité de 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.

Le processeur exécute deux cycles entre deux décisions, sauf si le processus termine avant. De 0 à 2, P1 exécute deux cycles. À l'instant 2, P1 et P2 ont chacun 6 cycles restants ; P1, arrivé le premier, est choisi pour les deux cycles suivants, de 2 à 4. P3, arrivé à 3, attend donc la prochaine sélection et s'exécute de 4 à 6. P1 s'exécute ensuite de 6 à 8 puis, après la sélection à 8, de 8 à 10 : à cet instant, P1 et P4 ont chacun 2 cycles restants, et P1 est plus ancien. P4 s'exécute de 10 à 12, puis P2 termine de 12 à 18.

Ordonnancement par tranches de deux cycles

Intervalle de temps	Processus
0–4	P1
4–6	P3
6–10	P1
10–12	P4
12–18	P2

Les intervalles consécutifs de P1 sont regroupés lorsqu'il est sélectionné plusieurs fois de suite.

RÉPONSE À RETENIR

Le nouvel ordonnancement est : 0–4 : P1 ; 4–6 : P3 ; 6–10 : P1 ; 10–12 : P4 ; 12–18 : P2.

Question d.

Sujet p. 9

> 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 instants de terminaison sont 10 pour P1, 18 pour P2, 6 pour P3 et 12 pour P4. On soustrait à chacun son instant d'arrivée : P1, 10 − 0 = 10 ; P2, 18 − 2 = 16 ; P3, 6 − 3 = 3 ; P4, 12 − 7 = 5 cycles. La somme de ces temps est 34 cycles.

RÉPONSE À RETENIR

La nouvelle moyenne est $\frac{10+16+3+5}{4} = 8{,}5$ cycles. Elle est supérieure à la moyenne précédente, qui était de 8 cycles : selon le critère de la moyenne des temps d'exécution, ce nouvel ordonnancement n'est pas plus performant.

FormaV - document d'entraînement non officiel

3

Question a.

Sujet p. 9, 10

> Les processus sont représentés par les listes fournies, contenant leurs durées en nombre de cycles et auxquelles sont ajoutées en fin de liste des chaînes de caractères vides pour simuler leur date de création. Recopier sans les commentaires et compléter la fonction choix_processus afin qu'elle renvoie l'indice du processus dont le temps restant d'exécution est le plus court parmi les processus en liste d'attente.

Une fois les processus présents ajoutés à la liste d'attente, leurs chaînes vides d'arrivée future ont été retirées : la longueur de leur liste correspond donc au nombre de cycles restants. On initialise le minimum avec le premier processus, puis on parcourt les suivants. Le test strict < laisse l'indice du premier processus en cas d'égalité ; comme les processus sont rangés par instant d'arrivée, cela respecte la règle de départage. La fonction effectue un parcours linéaire de la liste d'attente, donc son nombre de comparaisons est proportionnel à sa longueur. Elle n'est appelée que lorsque cette liste n'est pas vide.

```
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
```

Trace avec les données du sujet : à l'instant 2, P1 et P2 ont chacun 6 cycles restants ; le test strict conserve l'indice 0, celui de P1. À l'instant 3, les longueurs de P1, P2 et P3 sont respectivement 5, 6 et 2 ; la fonction renvoie l'indice 2, celui de P3.

> **RÉPONSE À RETENIR**
>
> La fonction renvoie l'indice du processus ayant le moins de cycles restants. En cas d'égalité, elle conserve le premier processus de la liste, donc le plus ancien.

Question b.

Sujet p. 9, 10

> La fonction scrutation parcourt la liste liste_proc des processus et renvoie la liste d'attente en fonction de leur arrivée : les processus présents, sans chaînes de caractères vides en fin de liste, sont ajoutés à la liste d'attente ; pour les autres, une chaîne vide est supprimée. Recopier et compléter les instructions manquantes de la fonction ordonnancement pour réaliser le fonctionnement désiré.

À chaque tour, on choisit le processus de durée restante minimale, on ajoute à `execution` le premier caractère de son premier élément (par exemple `'1'` pour `'1.8'`), puis on retire cet élément, qui représente le cycle exécuté. Si sa liste devient vide, ce processus terminé est retiré de la liste d'attente. On suppose que `scrutation` ajoute les listes des processus nouvellement arrivés sans dupliquer les processus déjà en attente, et qu'elle n'ajoute pas un processus terminé. Chaque tour consomme un élément de durée ; avec les données du sujet, les 18 éléments de durée sont donc consommés et la boucle termine. Le choix est refait à chaque cycle, comme demandé.

```
def ordonancement(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
```

Trace avec les données du sujet : les choix successifs regroupés en intervalles sont 0–3 : P1 ; 3–5 : P3 ; 5–7 : P1 ; 7–9 : P4 ; 9–12 : P1 ; 12–18 : P2. Le choix est notamment réévalué à l'arrivée de P2 à 2, de P3 à 3 et de P4 à 7. Chaque tour enlève un cycle de la liste choisie ; la boucle se termine lorsque les processus ont tous consommé leurs durées.

> **RÉPONSE À RETENIR**
>
> La liste d'exécution est `['1', '1', '1', '3', '3', '1', '1', '4', '4', '1', '1', '1', '2', '2', '2', '2', '2', '2']`. Elle contient 18 éléments, un par cycle exécuté.

Sujet : https://www.sujetdebac.fr/annales-pdf/2023/spe-numerique-informatique-2023-polynesie-2-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.