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

Amérique du sud - novembre 2025 - sujet 1 (corrigé)

Exercice 1 (Bases de données, SQL, arbres binaires)
Partie A

1. L'attribut `masse` de la relation `Exoplanetes` ne peut pas servir de clé primaire de cette relation car il n'est pas unique : plusieurs exoplanètes peuvent avoir la même masse comme, par exemplen celles d'`id_exoplanete` 6 et 7.
2. L'attribut `id_exoplanete` peut être utilisé comme clé primaire dans la relation `Exoplanetes` car il est unique et permet donc d'identifier sans ambiguïté chaque exoplanète.
3. Cette requête renvoie la masse et le rayon de l'exoplanète d'identifiant 4, sous la forme de cette table :

masse	rayon
0.01	0.16

4. La requête suivante convient :

```
SELECT id_etoile, nom FROM Etoiles WHERE ascension > 100
```

5. La requête suivante convient :

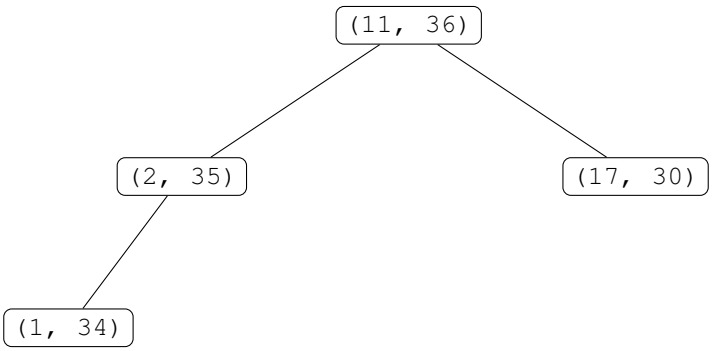
```
INSERT INTO Exoplanetes VALUES (9, 0.03, 0.37 ,4)
```

6. La requête suivante convient :

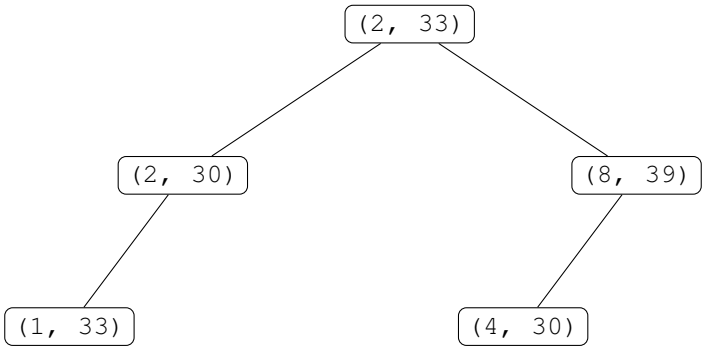
```
SELECT rayon FROM Exoplanetes
JOIN Etoiles
ON Exoplanetes.id_etoile = Etoiles.id_etoile
WHERE nom = 'Kepler-11'
```

Partie B

7. L'expression `sorted(etoiles)` vaut `[(10,30), (15,20), (17, 14), (29, 21), (30, 63), (35, 13)]`.
8. On a l'arbre binaire suivant :



9. On a l'arbre binaire de recherche suivant :



10. Le code suivant convient :

```
def construction(etoiles, debut, fin):
    if debut == fin:
        return None
    milieu = (debut + fin) // 2
    sag = construction(etoiles, debut, milieu)
    racine = etoiles[milieu]
    sad = construction(etoiles, milieu + 1, fin)
    return (sag, racine, sad)
```

11. La fonction suivante convient :

```
def en_arbre(etoiles):
    return construction(etoiles, 0, len(etoiles))
```

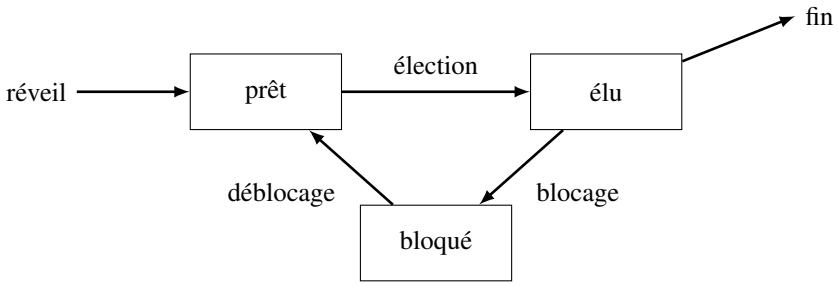
12. Le code suivant convient :

```
def contient(arbre, position):
    if arbre is None:
        return False
    sag, valeur, sad = arbre
    if position < valeur:
        return contient(sag, position)
    elif position == valeur:
        return True
    else:
        return contient(sad, position)
```

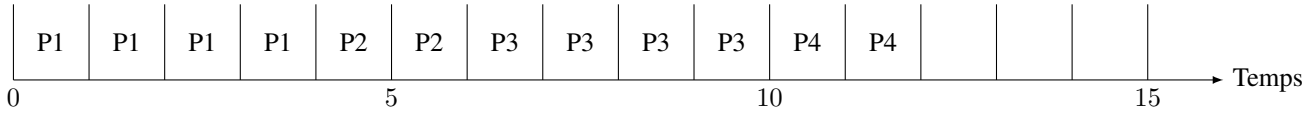
Exercice 2 (Systèmes d'exploitation, processus, POO)

Partie A

1. Un processus est un programme en cours d'exécution.
2. On a le schéma suivant :



3. La structure de données la plus adaptée pour gérer l'accès des processus au processeur selon la règle du « premier arrivé, premier servi » est la file, qui est basée sur le principe *First-In First-Out*.
4. On a le schéma suivant :



5. Le processus P4 arrive à l'instant 4 et n'est élu qu'à l'instant 10 donc il a attendu 6 unités de temps.

Partie B

6. L'interblocage est la situation où plusieurs processus sont bloqués car chacun attend qu'une ressource détenue par un autre se libère.
7. L'instruction suivante convient : `navigateurs = Priority_Queue()`
8. Le code suivant convient :

```
def sortir(self):
        """Retire et renvoie le dernier élément de
        liste_priorite"""
        assert not self.est_vide
        return self.liste_priorite.pop()
```

9. Le coût en temps de la recherche dichotomique est logarithmique en la taille de la liste.
10. Le code suivant convient :

```
def index_insertion_element(self, element):
        """Renvoie la position/index d'insertion
           d'element dans liste_priorite triée
           par ordre décroissant
           de numéro de priorité
        """
        if self.est_vide():
            return 0
        else:
            debut = 0
            fin = len(self.liste_priorite) - 1
            milieu = (debut + fin) // 2
            while debut <= fin:
                if self.liste_priorite[milieu][0] > element[0]:
                    debut = milieu + 1
                elif self.liste_priorite[milieu][0] < element[0]:
                    fin = milieu - 1
                else:
                    # cas d'égalité de priorité
                    return milieu
                milieu = (debut + fin) // 2
            return milieu + 1
```

11. La méthode suivante convient :

```
def inserer(self, element):
        """Modifie liste_priorite en insérant
           element à la position adéquate
           dans l'ordre décroissant de
           numéro de priorité"""
         index = self.index_insertion_element(element)
         self.liste_priorite.append(element)
         i = len(self.liste_priorite) - 1
         while i > index:
             self.liste_priorite[i] = self.liste_priorite[i-1]
             i = i -1
         liste_priorite[index] = element
```

Exercice 3 (Systèmes d'exploitation, réseaux et programmation Python)
Partie A

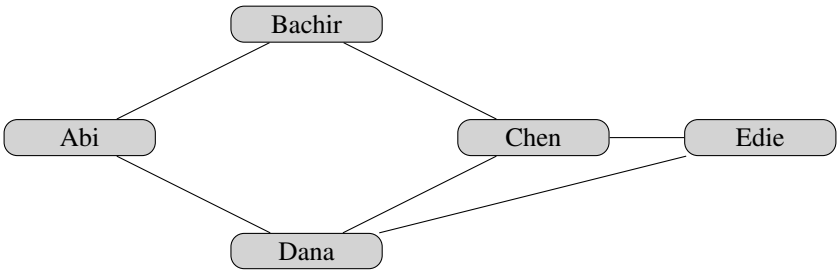
1. Le résultat de la commande `ls /association` est `adherents  annonces`
2. Il s'agit de la commande `cd annonces`
3. La commande `cp rosier_abi.html ../adherents/abi/` crée un nouveau fichier `/association/adherents/abi/rosier_abi.html` sans détruire `/association/annonces/rosier_abi.html`, tandis que `mv rosier_abi.html ../adherents/abi/` déplace le fichier ; dans le premier cas on a deux copies du fichier, dans le deuxième cas une seule.
4. On complète la section de code que Bachir doit modifier pour répondre à l'annonce :

```
<h2>En échange de</h2>
<table>
<tr><td>Nom :</td><td>Bachir</td></tr>
<tr><td>Plante :</td><td>Orchidée noire Cymbidium</td></tr>
<tr><td>Lien : </td><td><a href="monsiteperso.fr/bchir/orchidee.jpg">Photo de l'orchidée</a></td></tr>
</table>
```

5. Si Bachir exécute les étapes 1 et 2, puis que Chen exécute les étapes 1 à 3, et enfin que Bachir l'étape 3, alors les modifications de Chen sont supprimées.
6. On peut proposer ce nouvel algorithme :
   - Etape 1 : la personne déplace l'annonce vers son répertoire personnel ;
   - Etape 2 : la personne ouvre, modifie et enregistre le fichier dans son répertoire personnel ;
   - Etape 3 : le fichier modifié est déplacé vers le répertoire personnel de la personne qui a proposé l'annonce.

Partie B

7. Il suffit d'ajouter l'arête Dana-Edie :



8. On a la table de routage suivante :

Table de routage de Frida		
Destinataire	Intermédiaire	Distance
Abi	Abi	1
Bachir	Abi	2
Chen	Abi	3
Dana	Abi	2
Edie	Abi	3

9. Abi, Bachir, Chen, Dana et Edie doivent ajouter la ligne suivante dans leur table de routage :

Destinataire	Intermédiaire	Distance
Frida	Abi	$d+1$

où d est la distance avec Abi dans leur table de routage.

10. Voici les lignes qui doivent être modifiées.

Table de routage d'Abi		
Destinataire	Intermédiaire	Distance
Chen	Frida	2
Guy	Frida	2

Table de routage de Frida		
Destinataire	Intermédiaire	Distance
Hakim	Abi	2

Partie C

11. D'après les tables de routage, on a les affirmations suivantes :
    - ⋆ Hakim et Janus sont amis : vrai ;
    - ⋆ Hakim et Ines sont amis : faux ;
    - ⋆ Janus et Ines sont amis : vrai.
12. On obtient le code suivant :

```
def amis(table):
    """renvoie la liste des intermédiaires de la table de routage, sans doublon"""
    liste = []
    for (intermediaire, distante) in table.values():
        if intermediaire not in liste:
            liste.append(intermediaire)
    return liste
```

13. Le test de la ligne 5 du code de la fonction `maj` permet de vérifier si l'`ami` est absent de la table.
14. Le code de la ligne 6 est la suivante :

```
ma_table[ami] = (ami , 1)
```

15. Les codes des lignes 11 et 12 sont les suivantes :

```
if ma_table[adh][1] > distance + 1:
    ma_table[adh] = (ami, distance + 1)
```

16. La fonction suivante convient :

```
def nettoie(table):
    """Supprime toute les noms qui ne sont pas joignables"""
    nettoyage = []
    for (adh, ligne) in table.items():
        if ligne[0] == None:
            nettoyage.append(adh)
    for adh in nettoyage:
        del table[adh]
```

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.