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

25NSIJ1AS1

Exercice 1 (6 points)

Cet exercice porte sur les bases de données et les requêtes SQL, les arbres binaires et les algorithmes sur les arbres binaires.

Partie A

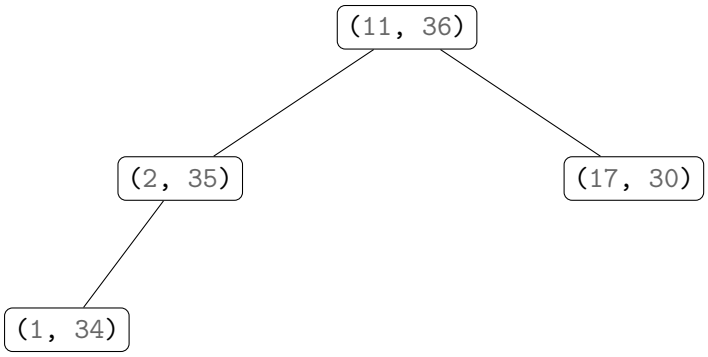
1. L'attribut masse de la relation `Exoplanetes` ne peut pas servir de clé primaire de cette relation car il n'est pas unique, plusieurs exoplanètes peuvent avoir la même masse, p.ex celles d'`id_exoplanete` 6 et 7.
2. L'attribut `id_exoplanete` peut être utilisé comme clé étrangère dans la relation `Exoplanetes` car il est unique et permet donc d'identifier sans ambiguïté chaque exoplanète.
3. La requête SQL `SELECT masse, rayon FROM Exoplanetes WHERE id_exoplanete = 4 ;` donne la masse et le rayon de l'exoplanète d'identifiant 4, sous la forme de cette table :

masse	rayon
0.01	0.16

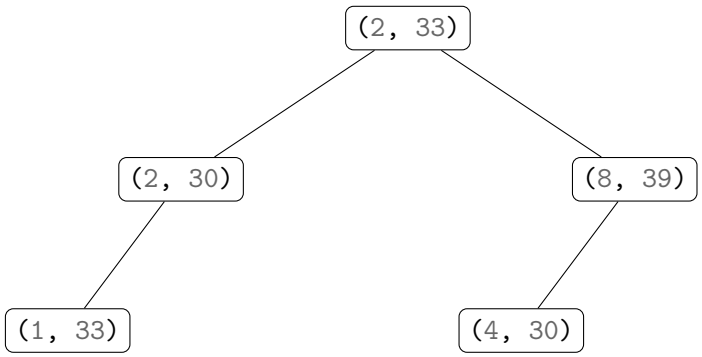
4. La requête `SELECT id_etoile, nom FROM Etoiles WHERE ascension > 100 ;` permet d'obtenir l'identifiant et le nom des étoiles dont l'ascension droite est supérieure ou égale à 100 degrés.
5. La requête `INSERT INTO Exoplanetes VALUES (9,0.03,0.37,4) ;` permet d'insérer la nouvelle exoplanète dans la base de données.
6. La requête `SELECT rayon FROM Exoplanetes NATURAL JOIN Etoiles WHERE nom = 'Kepler-11' ;` permet d'obtenir les rayons des exoplanètes orbitant autour de l'étoile nommée Kepler-11.

Partie B

7. L'expression `sorted(etoiles)` vaut `[(10,30), (15,20), (17, 14), (29, 21), (30, 63), (35, 13)]`.
8. le tuple `(((None, (1, 34), None), (2, 35), None), (11, 36), (None, (17, 30), None))` représente l'arbre binaire ci-dessous.



9. L'arbre binaire de recherche obtenu à partir de la liste `[(1, 33), (2, 30), (2, 33), (4, 30), (8, 39)]` est



10. On complète les lignes 3, 7, 8, 9 et 11 du code de la fonction `construction`.

```
1  def construction(etoiles, debut, fin):
2      if debut == fin:
3          return None
4
5      milieu = (debut + fin) // 2
6
7      sag = construction(etoiles, debut, milieu)
8      racine = etoiles[milieu]
9      sad = construction(etoiles, milieu + 1, fin)
10
11     return (sag, racine, sad)
```

11. On écrit le code de la fonction `en_arbre`.

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

12. On complète les lignes 3, 8, 9, 10 et 12 du code de la fonction `contient`.

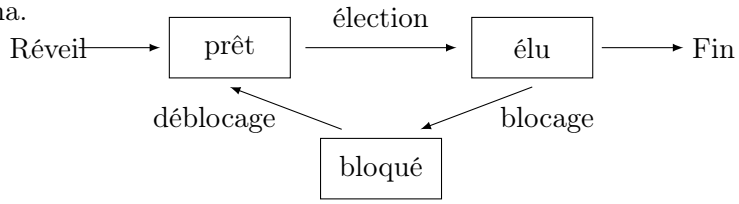
```
1  def contient(arbre, position):
2      if arbre is None:
3          return False
4
5      sag, valeur, sad = arbre
6
7      if position < valeur:
8          return contient(sag, position)
9      elif position == valeur:
10         return True
11     else:
12         return contient(sad, position)
```

Exercice 2 (6 points)

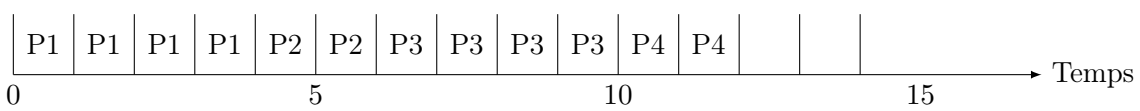
Cet exercice porte sur les systèmes d'exploitation, les processus, les structures de données linéaires, la programmation en Python et en particulier la programmation orientée objet.

Partie A

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



3. Donner 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 complète le schéma avec la règle du « premier arrivé premier servi ».



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 est `navigateurs=Priority_Queue()`.
8. On compléter la méthode `sortir`.

```
    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. On complète la méthode `index_insertion_element`.

```
    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. On écrit une méthode `inserer`.

```
    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 (8 points)

Cet exercice porte principalement sur les systèmes d'exploitation, les réseaux et la programmation de base en Python.

Partie A

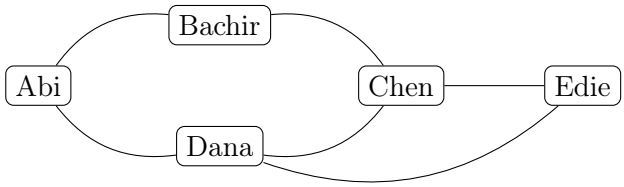
1. Le résultat de la commande ls /association est `adherents annonces`.
2. La commande `cd annonces` permet de définir le répertoire `annonces` comme répertoire courant à partir du répertoire `/association`.
3. La commande `cp rosier_abi.html ../adherents/abi/` crée un nouveau fichier `/association/adherents/abi/rosier_abi.html` 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<
</table>
</body></html>
```

5. Si Bachir exécute les étapes 1 et 2 puis que Chen exécute les étapes 1 à 3 ensuite 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. On reproduit le graphe en ajoutant l'arête Dana-Edie.



8. On complète la table de routage de Frida.

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

où d est la distance avec Abi dans leur table de routage, à laquelle on ajoute un.

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. On évalue chacune des affirmations.
    - Hakim et Janus sont amis : Vrai,
    - Hakim et Ines sont amis : Faux,
    - Janus et Ines sont amis : Vrai.

12. On remet les lignes de codes dans le bon ordre.

```
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
```

```
def maj(ma_table, ami, table_ami):
    """mise à jour de ma_table (dict) avec les
    informations de table_ami (dict).
    ami est du type str."""

    if ami not in ma_table.keys():
        ma_table[ami] = ( ... , ... )

    for adh in table_ami.keys():
        (intermediaire, distance) = table_ami[adh]
        if adh not in ma_table.keys():
            ma_table[adh] = (intermediaire, distance + 1)
        if ma_table[adh][...] > distance + 1:
            ma_table[adh] = (ami, ...)

    for adh in ma_table.keys():
        (intermediaire, distance) = ma_table[adh]
        if adh not in table_ami.keys() and adh != ami and intermediaire == ami:
            ma_table[adh] = (None, float('inf'))

    return ma_table
```

13. Le test de la ligne 6 du code de la fonction `maj` permet de vérifier si l'`ami` est absent de la table.
14. On complète la ligne 7 du code de la fonction `maj`.

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

15. On complète les lignes 13 et 14 du code de la fonction `maj`.

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

16. On propose une version corrigée de la fonction `nettoie`.

```
1  def nettoie(table):
2      """Supprime toute les noms qui ne sont pas joignables"""
3      nettoyage = []
4      for (adh, ligne) in table.items():
5          if ligne[0] == None:
6              nettoyage.append(adh)
7      for adh in nettoyage:
8          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.