

[← Index des sujets 2023](#)

23-NSIZERO-A : Corrigé

Année : **2023**

Centre : **Sujet Zéro-A**

Jour : x

Enoncé : [!\[\]\(e3f8612927870f2e0f9f5989e6dd3064_img.jpg\) PDF](#)

1. Exercice 1 (3 points)

Bases de données et types construits de données

1. Les attributs de la table `groupes` sont `idgrp`, `nom`, `style` et `nb_pers`.
2. L'attribut `nom` de la table `musiciens` ne peut pas être une clé primaire car plusieurs musiciens peuvent porter le même nom, or une clé primaire doit identifier de façon unique un enregistrement.
3. Cette requête renvoie la colonne `nom` de la table `groupes` lorsque le style du groupe est '`Latin Jazz`'. Sur l'extrait fourni, on obtient donc :

| nom |
|---------------------|
| 'Weather Report' |
| 'Return to Forever' |

1.  **Requête SQL**

```
UPDATE concerts
SET heure_fin = '22h30'
WHERE idconc = 36;
```

2. Pour récupérer le nom de tous les groupes qui jouent sur la scène 1 :

 **Requête SQL**

```
SELECT groupes.nom
FROM groupes
JOIN concerts ON groupes.idgrp = concerts.idgrp
WHERE concerts.scene = 1
```

3.  **Requête SQL**

```
INSERT INTO groupes
VALUES 15, 'Smooth Jazz Fourplay', 'Free Jazz', 4
```

4. Script Python

```
def recherche_nom(musiciens):
    au_moins_4_concerts = []
    for musicien in musiciens:
        if musicien['nb_concerts']>=4:
            au_moins_4_concerts.append(musicien[ 'nom' ])
    return au_moins_4_concerts
```

2. Exercice 2 (3 points)

Architecture matérielle, Réseaux et systèmes d'exploitation

1. L'adresse réseau de la configuration d'Alice est 172.16.2.0/24 , donc tous les ordinateurs de cette configuration ont une adresse IP qui commence par les 24 même premiers bits (i.e. les 3 premiers octets) : 172.16.2 . Cette configuration appartient donc à l'ordinateur d'Alice.
2. On applique la formule donnée dans l'énoncé avec un débit du réseau de 1 000 Mbits/s :
$$\text{cout} = \frac{10\ 000}{1\ 000} = 10$$
Le cout du réseau WAN8 est donc 10.
3. Voici la table du routeur R6 :

| Destination | Pass. | Cout |
|-------------|-------|------|
| LAN 1 | R5 | 21 |
| LAN 2 | - | - |
| WAN 1 | R5 | 11 |
| WAN 2 | R5 | 20 |
| WAN 3 | R5 | 11 |
| WAN 4 | R5 | 12 |
| WAN 5 | R5 | 10 |
| WAN 6 | - | - |
| WAN 7 | - | - |

| Destination | Pass. | Cout |
|-------------|-------|------|
| WAN 8 | R5 | 10 |

1. Les routeurs traversés seront : R6 → R5 → R2 → R1
2. Le routeur en panne est le routeur `R5` (car la nouvelle route est alors R6 → R4 → R2 → R1 dont le coût est bien 111.)

3. Exercice 3 (6 points)

ABR et programmation objet

1. Ce fonctionnement traduit le comportement d'une **file** c'est à dire que le premier élément qui entre dans la structure de données est aussi le premier à en sortir (*FIFO* pour *First In, First Out*). Dans une pile, la dernier élément entré est le premier à sortir (*LIFO* pour *Last In, First Out*)
2. a. C'est la **taille** de l'arbre (c'est à dire le nombre total de noeuds de l'arbre)
 b. C'est la **racine** de l'arbre puisqu'il s'agit de la tâche ajoutée à l'arbre en premier
 c. C'est une **feuille** de l'arbre aucune tâche n'ayant été ajouté après celle-ci, le noeud représentant cette tâche n'a pas de fils, c'est donc une feuille.
3. a. Les attributs de la classe `Noeud` sont `tache`, `indice`, `gauche` et `droite`.
 b. La méthode `insere` est récursive car elle s'appelle elle-même. Elle se termine car à chaque appel on descend d'un niveau dans l'arbre.
 c. On insère à gauche lorsque la valeur à insérer est inférieure à celle du noeud courant donc on complète la ligne 26 par :

 Script Python

```
elif self.racine.indice > nouveau_noeud.indice
```

d.

 État initial

```
graph TD
N("12") --> Ng("6")
N --> Nd("14")
Ng --> Ngg(" ")
Ng --> Ngd("10")
Ngd --> Ngdg("8")
Ngd --> Ngdd(" ")
Nd --> Ndg("13")
Nd --> Ndd(" ")
style Ngg fill:none, stroke-width:0px
style Ngdd fill:none, stroke-width:0px
style Ndd fill:none, stroke-width:0px

linkStyle 2 stroke-width:0px
linkStyle 5 stroke-width:0px
linkStyle 7 stroke-width:0px
```

 Après l'insertion de 11

```
graph TD
N("12") --> Ng("6")
N --> Nd("14")
Ng --> Ngg(" ")
Ng --> Ngd("10")
Ngd --> Ngdg("8")
Ngd --> Ngdd("11")
Nd --> Ndg("13")
Nd --> Ndd(" ")
style Ngg fill:none, stroke-width:0px
style Ndd fill:none, stroke-width:0px

linkStyle 2 stroke-width:0px
linkStyle 7 stroke-width:0px
```

 Après l'insertion de 5

```
graph TD
N("12") --> Ng("6")
N --> Nd("14")
Ng --> Ngg("5")
Ng --> Ngd("10")
Ngd --> Ngdg("8")
Ngd --> Ngdd("11")
Nd --> Ndg("13")
Nd --> Ndd(" ")

style Ndd fill:none, stroke-width:0px
linkStyle 7 stroke-width:0px
```

 Après l'insertion de 16

```
graph TD
N("12") --> Ng("6")
N --> Nd("14")
Ng --> Ngg("5")
Ng --> Ngd("10")
Ngd --> Ngdg("8")
Ngd --> Ngdd("11")
Nd --> Ndg("13")
Nd --> Ndd("16")
```

 Après l'insertion de 7

```
graph TD
N("12") --> Ng("6")
N --> Nd("14")
Ng --> Ngg("5")
Ng --> Ngd("10")
Ngd --> Ngdg("8")
Ngd --> Ngdgg("7")
Ngdgg --> Ngdgd(" ")
Ngdgd --> Ngdd("11")
Nd --> Ndg("13")
Nd --> Ndd("16")

style Ngdgd fill:none, stroke-width:0px
linkStyle 6 stroke-width:0px
```

a.  Script Python

```

41 def est_present(self, indice_recherche):
42     """renvoie True si l'indice de priorité indice_recherche (int) passé en paramètre
43 est déjà l'indice d'un noeud de l'abre, False sinon"""
44     if self.racine == None:
45         return False
46     else:
47         if self.racine.indice == indice_recherche:
48             return True
49         elif self.racine.indice > indice_recherche:
50             return self.racine.gauche.est_present()
51         else:
52             return self.racine.droite.est_present()

```

4. a. On rappelle que dans un parcours *infixe*, on parcourt le sous arbre gauche, puis la racine, puis le sous arbre droit. Dans le cas de l'arbre de la figure 1, on obtient : 6 → 8 → 10 → 12 → 13 → 14

b. La tâche la plus prioritaire sera le premier élément rencontré lors de ce parcours.

5. Script Python

```

1 def tache_prioritaire(self):
2     """renvoie la tache du noeud situé le plus à gauche de l'ABR supposé non vide"""
3     if self.racine.gauche.est_vide():
4         return self.racine.tache
5     else:
6         return self.racine.gauche.tache_prioritaire()

```

6.

Étape 1 : ajout de 14

```
graph TD
N("14")
```

Étape 2 : ajout de 11

```
graph TD
N("14")
N --> Ng("11")
N --> Nd(" ")
style Nd fill:none, stroke-width:0px
linkStyle 1 stroke-width:0px
```

❶ Étape 3 : ajout de 8

```
graph TD
N("14")
N --> Ng("11")
N --> Nd(" ")
Ng --> Ngg("8")
Ng --> Ngd(" ")
style Nd fill:none, stroke-width:0px
linkStyle 1 stroke-width:0px
style Ngd fill:none, stroke-width:0px
linkStyle 3 stroke-width:0px
```

❷ Étape 4 : traiter 8 qui est prioritaire

```
graph TD
N("14")
N --> Ng("11")
N --> Nd(" ")
style Nd fill:none, stroke-width:0px
linkStyle 1 stroke-width:0px
```

❸ Étape 5 : ajout de 12

```
graph TD
N("14")
N --> Ng("11")
N --> Nd(" ")
Ng --> Ngg(" ")
Ng --> Ngd("12")
style Nd fill:none, stroke-width:0px
linkStyle 1 stroke-width:0px
style Ngg fill:none, stroke-width:0px
linkStyle 2 stroke-width:0px
```

❹ Étape 6 : traiter 11 qui est prioritaire

```
graph TD
N("14")
N --> Ng("12")
N --> Nd(" ")
style Nd fill:none, stroke-width:0px
linkStyle 1 stroke-width:0px
```

❶ Étape 7 : traiter 12 qui est prioritaire

```
graph TD  
N("14")
```

❷ Étape 8 : ajout de 15

```
graph TD  
N("14")  
N --> Ng(" ")  
N --> Nd("15")  
style Ng fill:none, stroke-width:0px  
linkStyle 0 stroke-width:0px
```

❸ Étape 9 : ajout de 19

```
graph TD  
N("14")  
N --> Ng(" ")  
N --> Nd("15")  
Nd --> Ndg(" ")  
Nd --> Ndd("19")  
style Ng fill:none, stroke-width:0px  
linkStyle 0 stroke-width:0px  
style Ndg fill:none, stroke-width:0px  
linkStyle 2 stroke-width:0px
```

❹ Étape 10 : traiter 14 qui est prioritaire

```
graph TD  
N("15")  
N --> Ng(" ")  
N --> Nd("19")  
style Ng fill:none, stroke-width:0px  
linkStyle 0 stroke-width:0px
```