

[Index des sujets 2023](#)

## 23-NSIZERO-B : Corrigé

Année : 2023

Centre : Sujet Zéro-B

Jour : x

Enoncé : [PDF](#)

### 1. Exercice 1 (4 points)

Ligne de commande sous Linux, traitement de données en tables et bases de données

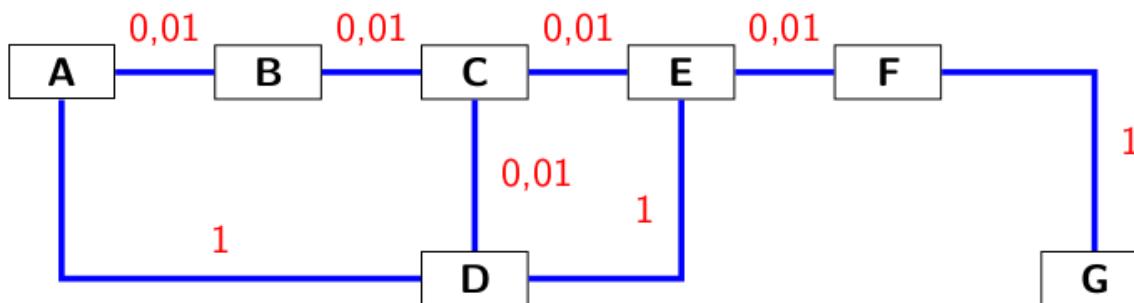
1. a. Les commandes permettant de se positionner dans `timbres` depuis `fiches` sont : commande **1** et commande **5**

b. Pour accéder au répertoire `timbres` depuis la racine, on peut écrire : `cd /home/document/collections/timbres`

2. a. On applique la formule  $C = \frac{10^8}{d}$  avec  $d = 100\text{Mbit/s}$  c'est à dire le coût d'une liaison FastEthernet :

$$C = \frac{10^8}{100 \times 10^6} = 1. \text{ Donc le coût d'une liaison FastEthernet est } 1.$$

b. On a reproduit ci-dessous le schéma du réseau, en faisant figurer le coût des liaisons (le coût d'une liaison FFTH est  $\frac{10^8}{10 \times 10^9} = 0,01$ ) :



Selon le protocole OSPF (minimisation des coûts), le chemin suivi sera donc : A → B → C → E → F → G pour un coût total de **1,04**.

3. Les descripteurs de ce fichier sont :

- `nom_timbre` avec pour valeurs `Gustave Eiffel`, `Marianne` et `Alan Turing`,
- `annee_fabrication` avec pour valeurs `1950`, `1989` et `2021`,
- `nom_collectionneur` avec pour valeurs `Dupont`, `Durand` et `Dupont`.

4. a. La clé primaire d'une relation est un attribut (ou ensemble d'attributs) permettant d'identifier de façon unique chaque enregistrement.
- b. L'attribut `nom` ne peut pas servir de clé primaire car il n'est pas unique pour chaque enregistrement. Dans l'exemple proposé plusieurs timbres ont pour nom `Gustave Eiffel`.
- c. Pour la même raison, l'attribut `annee_fabrication` ne peut pas servir de clé primaire non plus. Dans l'exemple proposé plusieurs timbres ont pour année de fabrication 1989.
- d. On peut ajouter un attribut `id_timbre` qui est différent pour chaque enregistrement (par exemple en l'incrémentant de 1 à chaque ajout d'un timbre)
5. a. Cette requête modifie l'attribut `ref_licence` en `Ythpswz` pour les enregistrements dont l'attribut `nom` est `Dupond`. Après cette requête la relation devient (en italique, les valeurs modifiées):

<code>ref_licence</code>	<code>nom</code>	<code>prenom</code>	<code>annee_naissance</code>	<code>nbre_timbres</code>
Hqdfapo	Dupuis	Daniel	1953	53
<i>Ythpswz</i>	Dupond	Jean-Pierre	1961	157
Qdfqnay	Zaoui	Jamel	1973	200
Aerazri	Pierre	Jean	1967	130
<i>Ythpswz</i>	Dupond	Alexandra	1960	61

- b. L'attribut `ref_licence` ne peut plus être une clé primaire puisqu'il est n'est plus unique (la valeur `Ythpswz` apparaît pour deux enregistrement)

## 6. Requête SQL

```
SELECT nom, prenom, nbre_timbres
FROM collectionneurs
WHERE annee_naissance >= 1963;
```

## 2. Exercice 2 (4 points)

Analyse et écriture de programmes récursifs

1. a. Une fonction récursive qui est une fonction qui s'appelle elle-même.
- b. La fonction `compte_rebours` ne fait rien si l'argument `n` passé en paramètre est négatif. Après l'affichage de `0`, `compte_rebours` est appelé avec la valeur `-1` et donc le programme s'arrête.

## 2. Script Python

```
def fact(n):
    """ Renvoie le produit des nombres entiers strictement positifs inférieurs à n """

```

```

if n == 0:
    return 1
else:
    return n * fact(n-1)

```

3. a. Dans la console l'affichage produit sera :

#### Console Python

```

3
2
1

```

En effet, `somme_entiers_rec(3)` va afficher 3 et appeler `somme_entiers(2)` qui va afficher 2 et appeler `somme_entiers(1)` qui va afficher 1.

b. La valeur 6 sera affecté à la variable `res` ( $3 + 2 + 1$ )

4.

#### Script Python

```

def somme_entiers(n):
    somme = 0
    for k in range(1,n+1):
        somme = somme + k
    return somme

```

## 3. Exercice 3 (4 points)

*ABR et POO*

1. a. Un exemple d'attribut de la classe `ArbreBinaire` est `valeur`. Un exemple de méthode est `insert_gauche`.  
b. a aura le valeur 15 et c la valeur 6.

2.

```

graph TD
S15["15"] --> S6["6"]
S15 --> S18["18"]
S6 --> S3["3"]
S6 --> S7["7"]
S18 --> S17["17"]
S18 --> S20["20"]
S3 --> S2["2"]
S3 --> V1[" "]
style V1 opacity:0;
linkStyle 7 stroke:#FFFFFF,stroke-width:0px

```

3. La valeur 13 figure dans le sous arbre gauche du noeud 12 qui ne doit contenir que des valeurs *plus petites* que 12 si l'arbre était binaire. On obtient un arbre binaire de recherche en inversant les positions du 13 et du 11.

```

graph TD
S10["10"] --> S3["3"]

```

```
S10 --> S12["12"]
S3 --> S2["2"]
S3 --> S5["5"]
S12 --> S11["11"]
S12 --> S13["13"]
```

4. La liste renvoyée sera : [1, 6, 10, 15, 16, 18, 25] . On rappelle que dans un parcours infixé, on parcourt d'abord le sous arbre gauche, puis la racine puis le sous arbre droit. Et que dans le cas d'un arbre binaire de recherche ce parcours permet d'obtenir les valeurs dans l'ordre croissant.