CORRECTION – NSI - 2023 Sujet Zéro B

Exercice 1 : Lignes de commandes et base de données

1. a. Les commandes possibles sont :
1 (avec un chemin absolu)
5 (avec un chemin relatif)

1. b. Chemin depuis la racine :
```
cd /home/documents/collections/timbres
```

2. a. Calcul du coût d'une liaison FastEthernet (BP = 100 Mbits/s) :

$C = \frac{10^8}{BP} = \frac{10^8}{100 \times 10^6} = 1$

2. b. Route : A→ B → C → E → F → G

3. Les descripteurs et leurs valeurs associées sont :
nom_timbre : Gustave Eiffel, Marianne, Alan Turin :
annee_fabrication : 1950 ,1989, 2012
nom_collectionneur : Dupont, Durant, Dupont

4. a. Une clé primaire est une donnée qui permet d'identifier de manière unique un enregistrement dans une table.

4. b. L'attribut nom ne peut pas jouer le rôle de clé primaire parce qu'un timbre (même nom) peut être édité plusieurs années.

4. c. L'attribut annee_fabrication ne peut pas être utilisée comme clé primaire car plusieurs timbres peuvent être fabriqués la même année.

4. d. Il faut utiliser un identifiant numérique incrémentée automatiquement à chaque ajout d'enregistrement (ex : id_timbre).

5. a. Cette requête met à jour la valeur de ref_licence avec `'Ythpswz'` pour les enregistrements dont l'attribut nom est `'Dupont'`.

5. b. Après cette mise à jour, l'attribut ref_licence ne peut plus être clé primaire parce que maintenant deux enregistrements possèdent la même valeur. (Le gestionnaire de base de donnée aurait du bloquer cette mise à jour car la contrainte de relation n'a pas été respectée.)

6.
```
SELECT nom, prenom, nbre_timbres
FROM collectionneurs
WHERE annee_naissance >= 1963
ORDER BY nom, prenom ;
```

Exercice 2 : Fonction récursives

1. a. Une fonction récursive est une fonction qui effectue au moins un appel à elle même.

1. b. Le programme s'arrête parce que la condition : n ≥ 0 devient fausse grâce au fait que les appels récursifs se font en décrémentant n.

2.
```
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. L'affichage : 3
2
1

3. b. La variable `res` a la valeur : 6 (3 + 2 + 1 = 6)

4.
```
def somme_entiers(n : int) -> int:
    """ Renvoie la sommes des nombres entiers
    strictement positifs inférieurs à n """
    s = 0
    for i in range(n+1):
        s += i
    return somme_entiers
```

ou

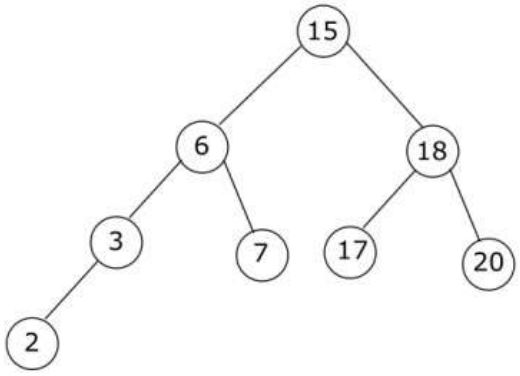
```
def somme_entiers(n : int) -> int:
    """ Renvoie la sommes des nombres entiers
    strictement positifs inférieurs à n """
    return sum(range(n+1))
```

Exercice 3 : Arbres binaire de recherche

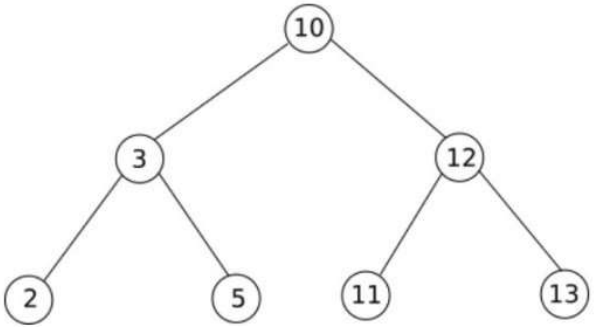
1. a. exemple d'attribut : `valeur`
exemple de méthode : `insert_droit()`

1. b. a = 15
c = 6

2.



3. Cet arbre n'est pas un arbre binaire de recherche car la valeur 13 est située en fils gauche du nœud 12 alors qu'il devrait être en fils droit.



4. La liste obtenue par le parcours infixe de l'arbre est : `[1, 6, 10, 15, 16, 18, 25]`