2022 Asie jour 2

Exercice 1

1.a.
- utilisateur : `gestion`
- ordinateur : `capNSI-ordinateur_central`

1.b. `ls Contrats`

2.a. `mkdir Contrats/TURING_Alan`

2.b. `chmod 774 Contrats/TURING_Alan`
(l'exemple dans l'annexe ne fonctionne pas)

3.

```
def formatage(tab):
    tab_ch = []
    for t in tab:
        tab_ch.append(t[0] + "_" + t[1])
    return tab_ch
```

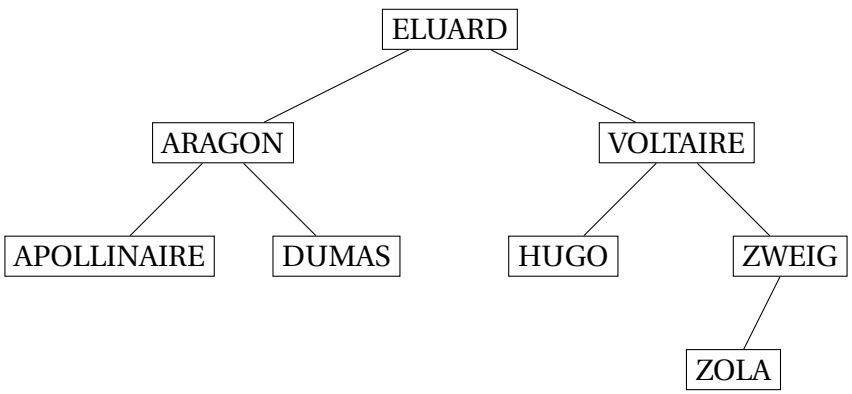
4.

```
import os

def creation_dossiers(tab):
    for ch in tab:
        dos = "Contrats/" + ch
        os.mkdir(dos)
        os.chmod(dos, 774)
```

Exercice 2

1.a.

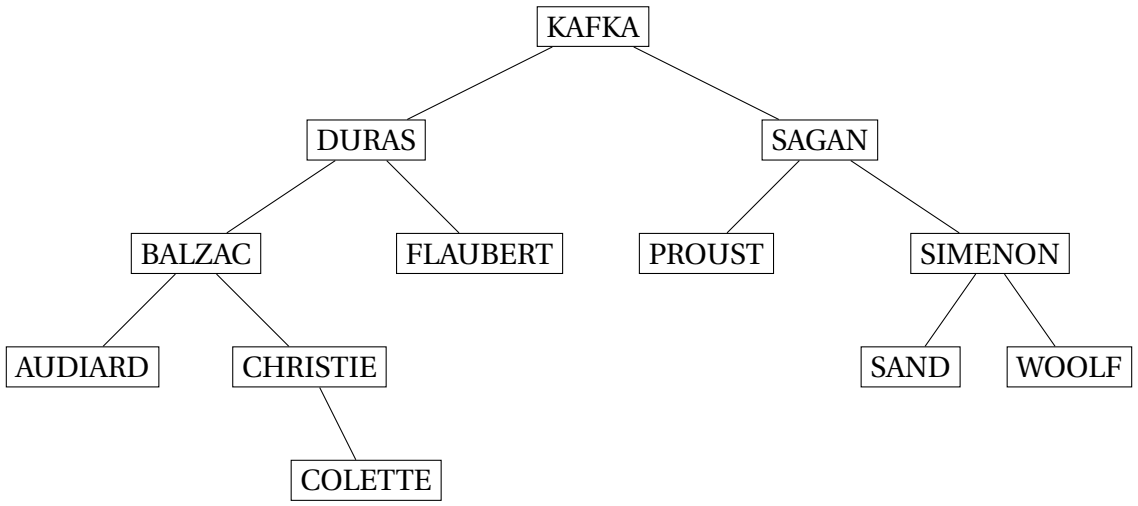


1.b.

— taille : 8
— hauteur : 4

1.c. $2^h - 1$

2.



3.

Il renvoie `True`.
Au premier appel, il appeler `mystere` sur le sous-arbre de racine `DURAS` (qui va renvoyer `FAUX` car ses deux fils sont `Null`) et sur le sous-arbre de racine `SAGAN` (qui va renvoyer `VRAI` car l'un de ses fils vaut `SIMENON`).

4.

```
Fonction hauteur(ABR) :
    SI ABR = NULL :
        RENVOYER 0
    SINON :
        RENVOYER 1 + MAX(hauteur(fils_gauche(ABR),hauteur(fils_droit(ABR))
```

Exercice 3

1.a. Le choix 2 est le plus adapté car dans le choix 1 c'est la même ligne qui est ajoutée à jeu. Ainsi lorsqu'on modifiera une ligne, toutes les lignes seront modifiées.

1.b. `jeu[5][2] = 1`

2.a.

```
import random

def remplissage(n, jeu):
    for i in range(n):
        x = random.randrange(8)
        y = random.randrange(8)
        # On cherche une cellule qui n'est pas déjà vivante
```

```
        while jeu[x][y] == 1:
            x = random.randrange(8)
            y = random.randrange(8)
        jeu[x][y] = 1
```

2.b. On doit avoir `0 <= n <= 64`

3.

```
def nombre_de_vivants(i, j, jeu):
    nb = 0
    voisins = [(i-1,j-1), (i-1,j), (i-1,j+1), (i,j+1),
               (i+1,j+1), (i+1,j), (i+1,j-1), (i,j-1)]
    for e in voisins :
        if 0 <= e[0] < 8 and 0 <= e[1] < 8 :
            nb = nb + jeu[e[0]][e[1]]
    return nb
```

4.

```
def transfo_cellule(i, j, jeu):
    nb_vivants = nombre_de_vivants(i, j, jeu)
    # règle 1
    if jeu[i][j] == 0:
        if nb_vivants == 3:
            return 1
        else:
            return 0
    # règle 2
    if jeu[i][j] == 1:
        if nb_vivants == 2 or nb_vivants == 3:
            return 1
        else:
            return 0
```

Exercice 4

1.a. `id_match` est la clé primaire de la relation `match`.

1.b. Oui elle en a quatre : `id_creneau`, `id_terrain`, `id_joueur1`, `id_joueur2`

2.a. C'est le premier août 2020 entre 10h et 11h.

2.b. Ce sont Dupont Alice et Durand Belina.

3.a. `SELECT prenom_joueur FROM joueurs WHERE nom_joueur = 'Dupont';`

3.b. `UPDATE joueurs SET mdp = 1976 WHERE id_joueur = 4;`

4.

```
INSERT INTO joueurs (id_joueur, nom_joueur, prenom_joueur, login, mdp)
VALUES (5, 'MAGID', 'Zora', 'zora', 2021);
```

On peut également ne pas nommer les champs :

```
INSERT INTO joueurs
VALUES (5, 'MAGID', 'Zora', 'zora', 2021);
```

5.

```
SELECT date
FROM matchs
JOIN joueurs
ON matchs.id_joueur1 = joueurs.id_joueur
OR matchs.id_joueur2 = joueurs.id_joueur
WHERE prenom_joueur = 'Alice';
```

Exercice 5

1. La fonction `range` va fournir des entiers entre `0` et `n-1`, on peut donc corriger la fonction somme de cette façon :

```
def somme(n) :
    total = 0
    for i in range(n) :
        total = total + 1/(i+1)
    return total
```

Une deuxième solution :

```
def somme(n) :
    total = 0
    for i in range(1, n+1) :
        total = total + 1/(i)
    return total
```

2.a. L'erreur déclenchée est un `IndexError` car `indice` peut prendre la valeur `len(L)` qui est trop grande. Voici une correction :

```
def maxi(L) :
    indice = 0
    maximum = 0
    while indice < len(L) :
        if L[indice] > maximum :
            maximum = L[indice]
        indice = indice + 1
    return maximum
```

2.b. Elle renvoie `0`. On peut corriger ce problème en initialisant le maximum avec la première valeur du tableau. On commencera alors avec l'indice `1` :

```
def maxi(L) :
    indice = 1
    maximum = L[0]
    while indice < len(L) :
        if L[indice] > maximum :
            maximum = L[indice]
        indice = indice + 1
    return maximum
```

3. Python ne peut pas concaténer une chaîne de caractères (`'Joueur'`) avec un entier (`i`). Il faut donc convertir l'entier en chaîne de caractères avec la fonction `str`.

```
def genere(n) :
    L = []
    for i in range(1, n+1) :
        L.append('Joueur ' + str(i))
    return L
```

4.a. 21

4.b. Il va y avoir une récursion infinie car la condition d'arrêt (`n == 0`) n'est jamais remplie (`n` passe de `1` à `-1` sans passer par `0`)

5.a. `(5, [10])`

5.b. `4, [10]`
Le tableau est quand même modifié ici car c'est sa référence qui est passée à la fonction.