Notes de correction

Exercice 1

1a. Nom utilisateur : `gestion` ; Nom de la machine : `capNSI-ordinateur_central`

1b. `cd Contrats ; ls` par exemple

2a. `cd Contrats` puis `mkdir TURING_Alan` ou `mkdir Contrats/TURING_Alan`

2b. `chmod rwx rwx r TURING_Alan`

3. une solution :

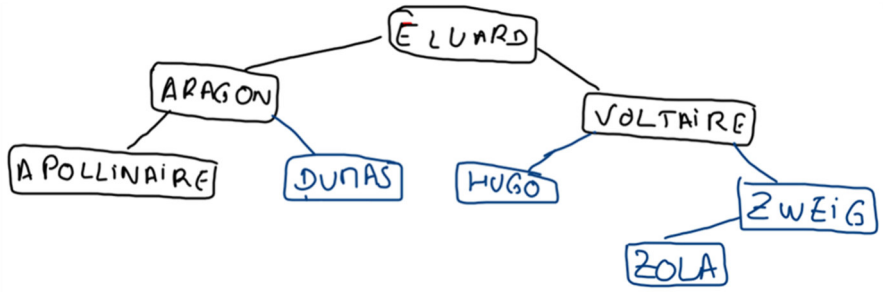
```
def formatage(tab) :
    l = []
    for couple in tab :
        l.append(couple[0]+'_'+couple[1])
    return l
```

4. Une solution :

```
import os
def creation_dossier(tab) :
    for t in tab :
        os.mkdir(t)
        os.chmod(t, 774)
```

Exercice 2

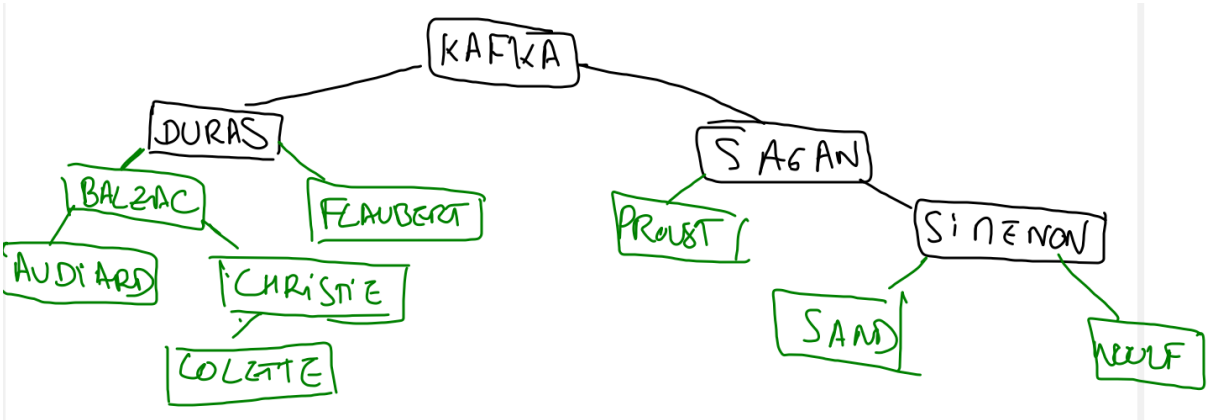
1a.



1b. Taille 8, hauteur 4

1c. Pour une hauteur h, il y a au maximum : $1 + 2 + 2^2 + 2^3 + \cdots + 2^{h-1} = 2^h - 1$ enregistrements possibles.

2.



3. L'appel renvoie VRAI, car la fonction réalise une recherche récursive du nom dans les étiquettes, le fils gauche et le fils droit.

4. Une solution :

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

Exercice 3

1a. La seconde solution est la bonne, car sinon on a un tableau de 8 fois la même ligne…. donc si on modifie une « case », on modifie toutes les lignes.

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

2a. une solution :

```
from random import randint
def remplissage(n, jeu) :
    nb = 0
    while nb < n :
        l = randint(0,7)
        c = randint(0,7)
        if jeu[l][c] == 0 :
            jeu[l][c] = 1
            nb = nb + 1
    return jeu
```

2b. il faut que n soit un entier naturel inférieur ou égal au nombre de 0 dans le tableau jeu.

3.

```
1 | def nombre_de_vivants(i, j, jeu) :
2 |     nb = 0
3 |     voisins = [(i-1,j-1), (i-1,j), (i-1,j+1), (i,j+1),
4 |                (i+1,j+1), (i+1,j), (i+1,j-1), (i,j-1)]
5 |     for e in voisins :
6 |         if 0 <= voisins[0] < 8 and 0 <= voisins[1] < 8 :
7 |             nb = nb + jeu[voisins[0]][voisins[1]]
8 |     return nb
```

4. Une solution :

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

Exercice 4

1a. clé primaire : `id_match`

1b. clé étrangère : `id_creneau, id_terrain, id_joueur1, id_joueur2`

2a. C'est le 1er Aout 2020 de 10h à 11h

2b. Dans le hangar, il s'agit de Dupont Alice contre Durand Belinda

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

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

4. `INSERT INTO joueurs (id_joueurs, nom_joueur, prenom_joueur, login, mdp) VALUES (5, 'MAGID', 'Zora', 'zora', 2021)`

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

Exercice 5

1. Il y a une division par 0 :

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

2a. Sortie de la liste :

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

2b. Mauvaise initialisation :

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

3 Impossible d'ajouter une chaîne avec un entier :

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

4a. `suite(6)` renvoie 21

4b. `suite(7)` ne renvoie rien : boucle infinie : `suite(7) = 3+2suite(5)` ……

5. Affichage :

```
(5, [10])
4 [10]
```