Éléments de correction sujet 02 (2022)

Exercice 1

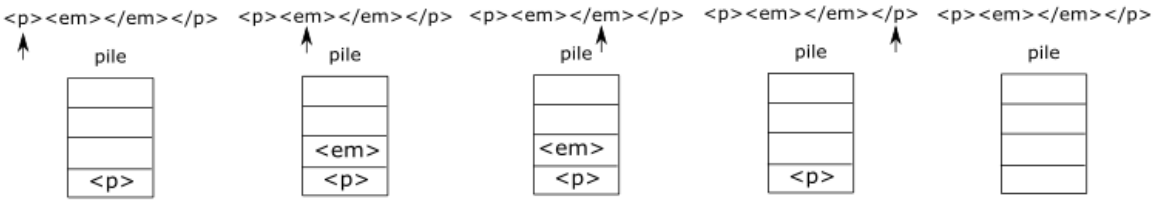
Partie A

1. Si les éléments sont retirés dans le même ordre qu'ils ont été ajoutés, cela décrit le comportement d'une file. La file se base sur le principe FIFO (First In First Out : premier entré, premier sorti).
2. B : 1, 2, 3, 2, 3, 2
   C : 1, 2, 1, 0, -1, 0
3.

```
def parenthesage_correct(expression) :
  controleur = 0
  for parenthese in expression:
    if parenthese == '(':
      controleur = controleur + 1
    else:
      controleur = controleur - 1
      if controleur < 0 :
        return False
  if controleur == 0:
    return True
  else:
    return False
```

Partie B

4. a.



   b. La séquence est correcte si la pile est vide après avoir parcouru l'ensemble de la séquence.

5. Puisque l'on a à chaque fois une balise ouvrante et une balise fermante, la pile pourrait au maximum contenir 6 éléments (12 divisé par 2).

Exercice 2

1.
   a. Cette requête renvoie le nom, le prénom et la date de naissance des personnes ayant pour nom ‘Crog’.
   b.

```
SELECT id_rea, titre
FROM realisation
WHERE annee > 2020
```

2.
   a. Il faudra utiliser la requête 1 puisque l’on cherche ici à effectuer une mise à jour (UPDATE). La requête 2 provoquera une erreur puisque cette requête va chercher à ajouter une nouvelle entrée avec une clé primaire pré-existante (688).
   b. La relation individu peut accepter deux individus portant le même nom, le même prénom, la même date de naissance à condition que la clé primaire (id_ind) soit différente pour les deux entrées.

3.
   a.

```
INSERT INTO emploi VALUES (5400, 'Acteur(James Bond)', 688, 105);
INSERT INTO emploi VALUES (5401, 'Acteur(James Bond)', 688, 325);
```

   b. Il est d’abord nécessaire de créer l’enregistrement du film dans la relation **realisation** puisque la nouvelle entrée dans la relation **emploi** devra faire référence au nouveau film 'Docteur Yes' dans la relation **realisation** (présence d’une clé étrangère dans la relation **emploi**).

4.
   a.

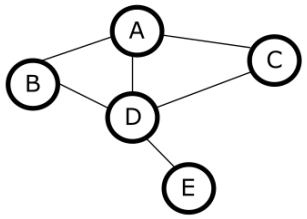
```
SELECT nom, titre, annee
FROM emploi
JOIN individu ON emploi.id_ind = individu.id_ind
JOIN realisation ON emploi.id_rea = realisation.id_rea
WHERE emploi.description = 'Acteur(James Bond)';
```

   b.

```
SELECT description
FROM emploi
JOIN individu ON emploi.id_ind = individu.id_ind
WHERE prenom = 'Denis' AND nom = 'Johnson';
```

Exercice 3

1.
   a. 192.168.128.131
   b. nombre d'adresses possible 256 (nombre de machines réellement adressables : 254, car il faut retirer le 0 (adresse réseau) et le 255 (adresse de broadcast)).
2.
   a. liste des routeurs directement reliés à A (métrique = 1) : B, C et D
   b.



3.

Débit	100 kbps	500 kbps	10 Mbps	100 Mbps
Métrique associée	1000	200	10	1

4.
   a. le chemin emprunté par un message pour aller de F à I est :
   F -> H -> J -> K -> I (coût de 13, tous les autres chemins possibles ont un coût supérieur à 13).
   b.

Destination	Métrique
F	0
G	8
H	5
I	13
J	6
K	8
L	11

   c.
   Trois routeurs permettent d'accéder à F directement I, H, et G. Si H tombe en panne, il resterait donc uniquement G et I. Or le coût de la liaison I -> F est de 20, ce qui rend préférable l'option de passer par G dans tous les cas de figure (par exemple, même pour aller de I à F, le chemin retenu serait I -> K -> J -> G -> F avec un coût de 19).

Exercice 4

Partie A

1. La somme de cet arbre binaire est 32 (7+6+4+3+9+2+1)
2. racine : A ; noeud : B ; feuille : C ; SAG : D ; SAD : E
3. Proposition C
4.

```
def somme(t):
    s = 0
    for v in t :
        s = s + v
    return s
```

5. La fonction *parcourir* permet d'obtenir un parcours en largeur

Partie B

6. Proposition D
7. somme(arbre) = valeur_racine + somme(SAG) + somme(SAD)
8.

```
def calcul_somme(arbre):
    if est_vide(arbre):
        return 0
    else :
       return valeur_racine(arbre) + calcul_somme(arbre_gauche(arbre)) +
calcul_somme(arbre_droit(arbre))
```

Exercice 5

1. Instruction 3 : `joueur1 = Joueur("Sniper", 319, "A")`
2. a.

```
def redevenir_actif(self):
        if not self.est_actif :
            self.est_actif = True
```

b.

```
def nb_de_tirs_recus(self):
        return len(self.liste_id_tirs_recus)
```

3. a. test 1
   b. si le tir est fratricide l'équipe perd 20 points
4.

```
def collecte_information(self,participant):
    if participant.equipe == self.equipe :
        for id in participant.liste_id_tirs_recus:
            if self.est_un_id_allie(id):
                self.incremente_score(-20)
            else:
                self.incremente_score(-10)
        if participant.est_determine():
            self.incremente_score(40)
```