Proposition de correction

Exercice 1

Partie A

Q1

1er lu = 1er enregistré : pile FIFO = file

Q2

B : 1, 2, 3, 2, 3, 2

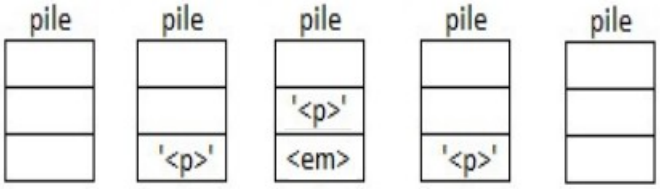
C : 1 2 1 0 -1 0

Q3

```
def parenthesage_correct(expression):
    """ fonction retournant True si l'expression arithmétique
    simplifiée (str) est correctement parenthésée, False sinon.
    Condition: expression ne contient que des parenthèses
    ouvrantes et fermantes """
    controleur = 0
    for parenthese in expression:      #pour chaque parenthèse
        if parenthese == '(':
            controleur = controleur + 1
        else:   # parenthese == ')'
            controleur = controleur - 1
            if controleur < 0 :      # test 1 (à recopier et compléter)
                #parenthèse fermante sans parenthèse ouvrante
                return False
    if controleur == 0 :          # test 2 (à recopier et compléter)
        return True     #le parenthésage est correct
    else:
        return False    #parenthèse(s) fermante(s) manquante(s)
```

Partie B

Q4.a



Q4.b

pile = vide

Q5

6 (nb balises / 2)

Exercice 2

Q1.a

Crog Daniel 07-07-1968

Q1.b

SELECT titre, id_rea

FROM realisation

WHERE annee > 2020

ORDER BY titre

Q2.a

requête 1

La requête 2 est une insertion et non une mise à jour : de plus, la clef primaire 688 existe déjà dans la table

Q2.b

il n'existe aucune contrainte de clef unique sur le triplet (nom, prénom, date de naissance)

Q3.a

- INSERT INTO emploi

  VALUES (5400, 'Acteur(James Bond)', 688, 105);
- INSERT INTO emploi

  VALUES (5401, 'Acteur(James Bond)', 688, 325);

Q3.b

La clef id_ind existe déjà dans la table individu

il faut que la clef id_rea existe au préalable dans la table realisation

Q4.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)';

Q5

SELECT description

FROM emploi

JOIN individu ON emploi.id_ind = individu.id_ind

WHERE individu.prenom = 'Denis' AND individu.nom = 'Johnson'

Exercice 3

Q1.a

192.168.128.131

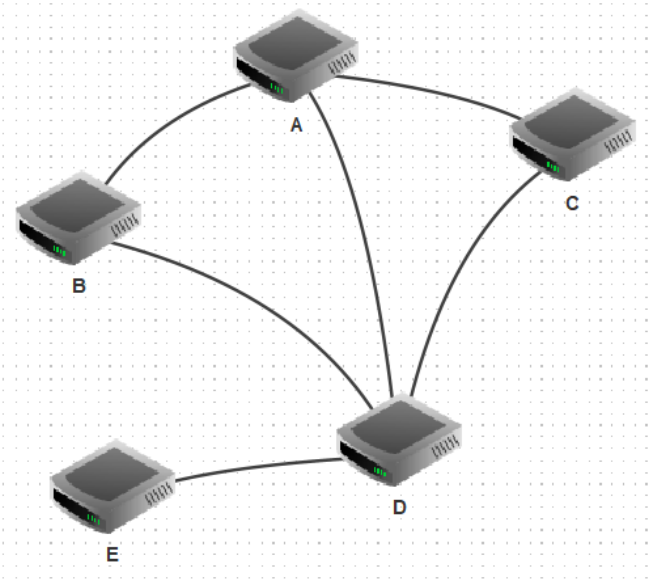
Q1.b

256 – 2 = 254

Q2.a

B, C, D

Q2.b



Q3

Débit	100 kbps	500 kbps	10 Mbps	100 Mbps
Métrique associée	1 000	200	10	1

Q4.a

F → H → J → K → I (total 13)

Q4.b

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

Q4.c

Rupture de la liaison F-H

La métrique la plus courte doit faire passer par G, sauf pour I → F

Exercice 4

Partie A

Q1

Parcours en largeur : 3 + 6 + 2 + 7 + 4 + 9 + 1 = 32

Q2

A : ‘racine’

C : ‘feuille

B : ‘nœud’

D : ‘SAG’

E : ‘SAD’

Q3

Proposition C

Q4

```
def somme(liste : list) -> int:
    """
    @param -- liste de nombres
    @return -- somme de ses éléments.
    """
    total = 0
    for i in liste:
        if type(i) is int:
```

```
        total += i
    return total
```

Q5

parcours en largeur

Partie B

Q6

Proposition D

Q7

S(arbre) = racine + S(SAG) + S(SAD)

S(arbre) = 3 + 17 + 12 = 32

Q8

```
def calcul_somme(arbre : object) -> int :
    if est_vide(arbre) :
            return 0
    return valeur_racine(arbre) + \
            calculs_somme(arbre_gauche(arbre)) + calculs_somme(arbre_droit(arbre))
```

Exercice 5

Q1

joueur1 = Joueur("Sniper", 319, "A")

Q2.a

```
def redevenir_actif(self):
    if self.est_actif == False:
        self.est_actif = True
```

Q2.b

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

Q3.a

Test #1

Q3.b

Il est doublement pénalisé -20 au lieu de -10

Q4

```
if participant.est_determine() :      #si le participant réalise un grand nombre de tirs
    self.incremente_score(40)         #le score de la Base augmente de 40
```