Proposition de correction

Exercice 1

Partie A

Q1

$2^8 - 2 = 254$

Q2

1101 1001

Q3

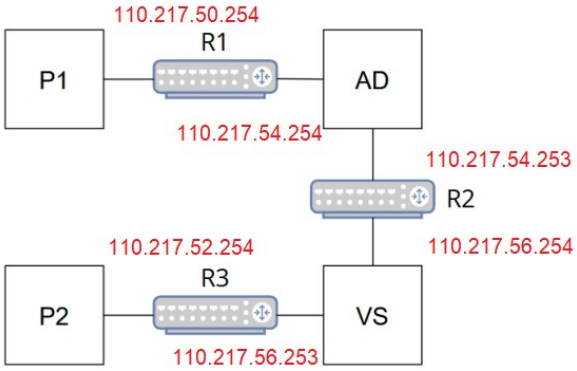
$0011\ 0010_2 = 32 + 16 + 2 = 50$

Q4

110.217.52.0/24 admet une plage @ de 110.217.52.0 à 110.217.52.255

Q5

Destination	Passerelle	Interface
110.217.50.0	on-link	110.217.50.254
110.217.52.0	110.217.54.253	110.217.54.254
110.217.54.0	on-link	110.217.54.254
110.217.56.0	110.217.54.253	110.217.54.254



Q6

Destination	Passerelle	Interface
110.217.50.0	on-link	110.217.50.254
110.217.52.0	110.217.50.253	110.217.50.254
110.217.54.0	on-link	110.217.54.254
110.217.56.0	110.217.54.253	110.217.54.254

Q7

non, le nombre de sauts reste inchangé pour R2

Partie B

Q8

La fonction effectue une recherche en profondeur mais ne garde pas trace des sommets déjà visités. Cela peut entraîner des boucles infinies si le graphe contient des cycles.

Q9

```
def recherche_v2(R1, R2, visités=None):
    if visités == None:
        visités = []       # évite les effets de bord !

    if R1 == R2:
        return True
    if R1 not in visités:
        visités.append(R1)

    for S in adjacents(R1,G):
        if S not in visités:
            if recherche_v2(S, R2, visités):
                return True

    return False
```

Exercice 2

Partie A

Q1

```
def possible_avec_penalites_seules(score : int) -> bool:
    """
    @param score -- un score
    @return True si score est un multiple de 3, False sinon
    """
    return not bool(score % 3)
```

Q2

Score	liste des solutions	nombre de solutions
0	[0]	1
1	[]	0
2	[]	0
3	[0, 3]	1
4	[]	0
5	[0, 5]	1
6	[0, 3, 6]	1
7	[0, 7]	1
8	[0, 5, 8], [0, 3, 8]	2
9	[0, 3, 6, 9]	1
10	[0,3,10], [0, 5, 10], [0,7,10]	3

Q3

Soit $f(n) = f(n-3) + f(n-5) + f(n-7)$

Soit $f(10) = f(10-3) + f(10-5) + f(10-7) = f(7) + f(5) + f(3) = 1 + 1 + 1 = 3$

Q4

n	$f(n)$
0	f(0) = 1
1	f(1) = 0
2	f(2) = 0
3	f(3) = 1
4	f(4) = 0
5	f(5) = 1
6	f(6) = 1

Q5

```
def nb_solutions(n : int) -> int:
    """
    @param n -- entier >= 0 correspondant à un score
    @return le nombre de possibilités d'obtenir ce score donné.
    """
    if n < 0:
        return 0
    elif n in [0, 3, 5, 6]:
        return 1
    elif n in [1, 2, 4]:
        return 0
    else:
        return nb_solutions(n - 3) + nb_solutions(n - 5) + nb_solutions(n – 7)
```

Q6

Memoization

Q7

F(11) = f(11−3) + f(11−5) + f(11−7) = f(8) + f(6) + f(4)

- f(4) []
- f(6) [0, 3, 6]
- f(8) [0, 5, 8], [0, 3, 8]

Q8

```
def solutions_possibles(score):
    if score < 0:
        resultat = []
    elif score == 0:
        resultat = [[0]]
    else:
        resultat = []
        for coup in [3, 5, 7]:
            liste = solutions_possibles(score - coup)
            for solution in liste:
                solution.append(coup + solution[len(solution)-1])
                resultat.append(solution)
    return resultat
```

Exercice 3

Partie A

Q1

- participants['PHILIPSEN Jasper']
- classement_general[participants['PHILIPSEN Jasper']]
- temps_etapes[participants['PINOT Thibaut']][3]

Q2

```
def calcul_temps_total(d : int) -> int:
    """
    @param d -- numéro d'un dossard d
    @return le temps total en seconde mis par ce coureur depuis le départ du tour de France
    """
    return sum(temps_etapes[d]) if d in temps_etapes else 0
```

Q3

```
classement = []

for numero_dossard in temps_etapes:
    element = (numero_dossard, calcul_temps_total(numero_dossard))
    classement.append(element)
    pos = len(classement) - 2

    while pos >= 0 and element[1] < classement[pos][1]:
        classement[pos + 1] = classement[pos]
        pos = pos - 1
        classement[pos + 1] = element

for i in range(len(classement)):
```

```
    classement_general[classement[i][0]] = i + 1
```

Q4

```
def classement_par_temps(tableau_temps):
    tableau_final = []
    difference_temps = 0
    premier = True
    for ligne in tableau_temps:
        coureur = [ligne[0]]
        coureur.append(ligne[1])
        if premier:
            temps_premier = ligne[2]
            coureur.append(temps_premier)
            premier = False
        else:
            difference_temps = ligne[2] - temps_premier
            coureur.append(difference_temps)
        tableau_final.append(coureur)
    return tableau_final
```

Partie B

Q5

Le tuple n° dossard + n° étape permet d'identifier de manière unique un enregistrement

Q6

la requête renvoie le nom des coureurs de l'équipe Cofidis

Q7

SELECT Date

FROM Etapes

WHERE Type = 'contre-la-montre'

ORDER BY Date

Q8

SELECT Equipes.directeurSportif

FROM Equipes

JOIN Coureurs

ON Equipes.nomEquipe = Coureurs.Equipe

WHERE nomCoureur = 'BARDET Romain'

Q9

la clé étrangère de valeur 5 n'existe pas encore dans la table Etapes et provoque une erreur d'intégrité référentielle

Q10

effectuer les requêtes dans cet ordre :

1. INSERT INTO Etapes VALUES(5, 'Montagne', 'Pau', 'Laruns', 05/07/2023);
2. INSERT INTO Temps VALUES (1, 5, 14267);

Q11

```
SELECT SUM(Temps.tempsRealise) AS totalTemps
FROM Temps
JOIN Coureurs
ON Temps.numDossard = Coureurs.numDossard
JOIN Etapes
ON Temps.numEtape = Etapes.numEtape
WHERE Coureurs.nomCoureur = 'BARDET Romain'
AND Etapes.date LIKE '2023-%-%'
```