Éléments de correction
sujet 06

Exercice 1

1. `balise12 = Balise(12, [“vert”, “noir”]`
2. `balise9.voisines = [balise8, balise11, balise12]]`
3. `[2, 5, 6]`
4. `[“noir”, “vert”]`
5.
```
def itineraire(balise_debut, balise_fin, couleur):
    assert couleur in balise_debut.couleurs_balise
    assert couleur in balise_fin.couleurs_balise
    balise = balise_debut
    chemin = [balise]
    while balise.num_balise != balise_fin.num_balise :
        for b in balise.voisines:
            if (couleur in b.couleurs_balise) and (b not in chemin):
                balise = b
                chemin.append(balise)
    return [b.num_balise for b in chemin]
```
6. 1, 2, 4, 5, 10, 7, 6, 3, 11, 9, 8, 12
7. `balise4.voisines = [(balise2, 13), (balise5, 21), (balise6, 15)]`
8. 5
9. 1, 2, 4, 6, 11, 9, 12
10.
```
def itineraire_trail(balise_debut, balise_fin):
    balise_debut.visitee = True
    balise = balise_debut
    chemin = [balise]
    while balise_fin not in chemin:
        prochaine = mystere(balise)
        if prochaine != None:
            prochaine.visitee = True
            chemin.append(prochaine)
            balise = prochaine
        else:
            return None
    return [b.num_balise for b in chemin]
```

11. algorithme glouton
12. positif : exécution en un temps raisonnable
    négatif : ne fournit pas forcément la meilleure solution

Exercice 2

1. v = 6
2.
```
def plateau_init(n, m, cartes):
    shuffle(cartes)
    plateau = []
    for i in range(n):
        plateau.append([cartes[i+j*n] for j in range(m)])
    return plateau
```
3.
```
def cartes_voisines(n, m, i, j):
    voisines = []
    for i2 in range(i-1, i+2):
        for j2 in (j-1, j+2):
            if (i2, j2) != (i, j) and i2 in range(n) and j2 in range(m):
                voisines.append((i2,j2))
    return voisines
```
4.

e1 = 9 + 8 + 6 + 7 = 30
e2 pas une chaine
e3 pas une chaine

5.
```
def chaine_value(plateau, chaine):
    s = 0
    for i,j in chaine:
        s += plateau[i][j]
    return s
```
6.

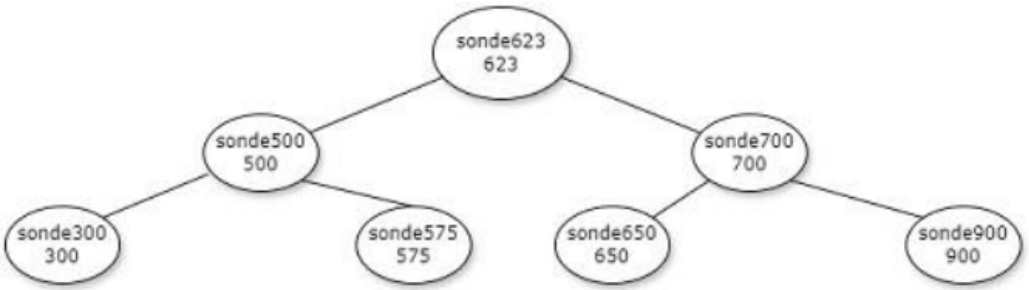
En faisant un parcours en largeur, la chaine qui aura la valeur recherchée sera forcément la plus courte.

7.
```
def explore(plateau, cible):
    n, m = len(plateau), len(plateau[0])
    a_visiter = file_init()
    for i in range(n):
        for j in range(m):
            file_ajoute(a_visiter, [(i,j)])
    while not file_est_vide(a_visiter):
        chemin = file_retire(a_visiter)
        if chaine_evalue(plateau, chemin) == cible:
            return chemin
        dernier = chemin[-1]
        i0, j0 = dernier
        for i, j in cartes_voisines(n, m, i0, j0):
            if (i, j) not in chemin:
                file_ajoute(a_visiter,chemin + [ (i,j) ])
    return None
```

8. On crée une nouvelle liste `tous_chemins` et à la ligne 11, au lieu de faire un `return chemin`, on fait un `tous_chemins.append(chemin)`

Exercice 3

1. `enregistrement['latitude']`
2. `('2024-06-27','23:36:01')`
3. 4 pour `len(frames)` et 5 pour `len(frames[1])`
4.
```
def detecter_anomalie(d):
    return d['altitude'] < 0 or d['altitude'] > 35000
```
5.
```
def liste_num_serie(frames):
    n_serie = []
    for e in frames:
        if e['num_serie'] not in n_serie:
            n_serie.append(e['num_serie'])
```
6.
```
def distance_totale(dep):
    total = 0
    for i in range(1,len(dep)):
        total += distance_haversine(dep[i-1], dep[i])
    return total
```
7.
```
sonde623 = Sonde(623,38.38825, 27.09004, '2024-06-27', None, None)
```
8.



9.
ordre infixe

10.
```
def rechercher(self, numero):
    if self.est_vide():
        return None
    if numero == self.num_serie:
        return self
    elif numero < self.num_serie:
        return self.gauche.rechercher(numero)
    else:
        return self.droit.rechercher(numero)
```
11. la fonction s’appelle elle-même
12. oui, id_abonne peut servir de clé primaire car il est unique pour chaque entrée
13. num_serie et id_abonne
14.

Détoile	Diane
Girard	Antoine

15.
```
INSERT INTO InfosRecuperation
VALUES
(14,480,24,'2024-07-10',47.230,12.244)
```
16.
```
SELECT sonde.num_serie, nom, date_recup
FROM info_recuperation
JOIN sonde ON sonde.num_serie = info_recuperation.num_serie
JOIN abonne ON abonne.id_abonne =  info_recuperation.id_abonne
```