Élément de correction
sujet 01

Exercice 1

1. Sorbier
2. Non, on ne peut pas l'identifier
3.
```
f1 = Feuille_resultat('Sorbier')
f2 = Feuille_resultat('Bobinier, Noyer')
f3 = Feuille_resultat('-')

n3 = Noeud("Bord denté ?", f1, f2)
n2 = Noeud('Alternées ?', n3, f3)
arbres_2 = Noeud('Simples ?', f3, n2)
```
4.
```
def est_resultat(self):
        return False
```
5.
```
def est_resultat(self):
        return True
```
6.
```
def nb_vegetaux(self):
        return len(self.vegetaux)
```
7.
```
def nb_vegetaux(self):
        return self.sioui.nb_vegetaux() + self.sinon.nb_vegetaux()
```
8.
```
def liste_questions(self):
        return []
```
9.
```
def liste_questions(self):
        return [self.question] + self.sioui.liste_questions() +
self.sinon.liste_questions()
```
10.
```
def est_bien_renseigne(dico_vegetal, arbre):
    quest = arbre.liste_questions()
    for q in quest :
        if q not in dico_vegetal:
            return False
    return True
```

11.

```
def identifier_vegetaux(dico_vegetal, arbre):
    while not arbre.est_resultat():
        if dico_vegetal[arbre.question]:
            arbre = arbre.sioui
        else :
            arbre = arbre.sinon
    return arbre.vegetaux
```

Exercice 2

1.

```
def passer_transit(self):
    self.etat = 'transit’
```

2.

```
def ajouter_colis(liste, colis):
    if colis.poids <= 25:
        liste.append(colis)
    else :
        print('Dépassement du poids maximum autorisé')
```

3.

```
def nb_colis(liste):
    return len(liste)
```

4.

```
def poids_total(liste):
    total = 0
    for c in liste:
        total = total + c.poids
    return total
```

5.

```
def liste_colis(liste, statut):
    lst = []
    for c in lst :
        if c.etat == statut:
            lst.append(c)
    return lst
```

6.

tri par sélection ; $O(n^2)$

7.

tri par insertion ; $O(n^2)$
ou tri fusion ; O(nlog(n))

8.
```
def chargement_glouton(liste, rang, capacite):
    if rang == len(liste):
        return []
    elif liste[rang].poids <= capacite:
        return [liste[rang]] + chargement_glouton(liste, rang+1,
capacite-liste[rang].poids)
    else :
        return chargement_glouton(liste, rang+1, capacite)
```

9.
On dépasse le nombre d'appels récursifs autorisés

10.
```
def chargement_glouton2(liste, capacite):
    colis_a_charger = []
    for c in liste :
        if c.poids <= capacite :
            colis_a_charger.append(c)
            capacite -= c.poids
    return colis_a_charger
```

Exercice 3

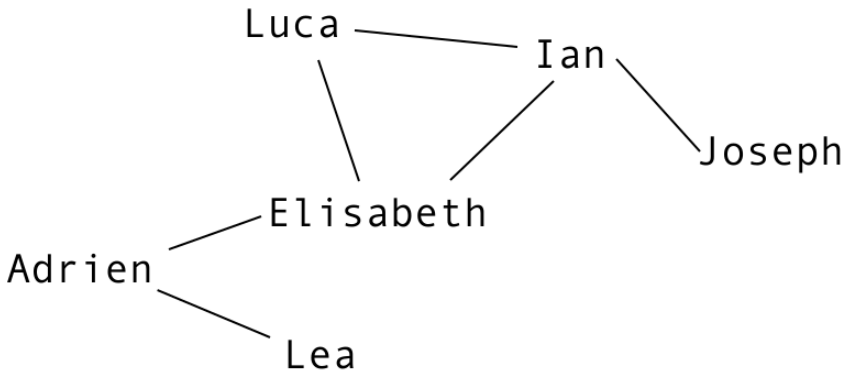
Partie A

1. Entier (INT)
2. On pourait avoir annee > 2007 et annee <= 2025
3. num_parent car reférence la table parent
4. tel car il n'est pas possible d'avoir 2 parents avec le même numéro (contrainte d'unicité)
5. parce que dans la table enfant, le numéro 33600782812 ne renvoie plus vers rien
6.
```
INSERT INTO parent VALUES ('Bauges', 33619782812, 73340);
UPDATE enfant SET num_parent = 33619782812 WHERE num_parent =
33600782812;
DELETE FROM parent WHERE tel =  33600782812
```
7.
Nakamura, Hawa, Kian, Adrien
8.
```
SELECT prenom
FROM enfant
WHERE num_parent = 3619861122
ORDER BY prenom
```

9.
```
SELECT id, prenom
FROM enfant
JOIN parent ON tel = num_parent
WHERE codep = 38520
```

Partie B

10.
parce que la mésentente est réciproque (A ne peut pas bien s'entendre avec B alors que B s'entend avec A)

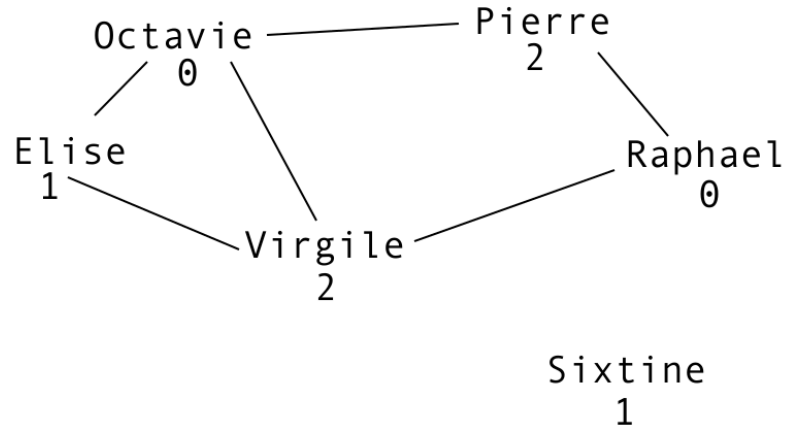
11.



12.
```
def degre(g,s):
    return len(g[s])
```

13.
```
def sommets_tries(g):
    sommets = [sommet for sommet in g]
    n = len(sommets)
    for i in range(1,n):
        sommet_courant = sommets[i]
        j = i-1
        while j >= 0 and degre(g,sommet[j]) <
degre(g,sommet_courant):
            sommets[j+1] = sommets[j]
            j = j - 1
        sommets[j+1] = sommet_courant
    return sommets
```

14.
tri par insertion ; quadratique

15.




16.

```
def colorer_graphe(g,dc):
    for s in dc:
        couleur = plus_petite_couleur_hors_voisins(g,dc,s)
        dc[s] = couleur
```

17.

```
def welsh_powell(g):
    dc = {}
    for s in g:
        dc[s] = -1
    for s in sommets_tries(g) :
        couleur = plus_petite_couleur_hors_voisins(g,dc,s)
        dc[s] = couleur
    return dc
```