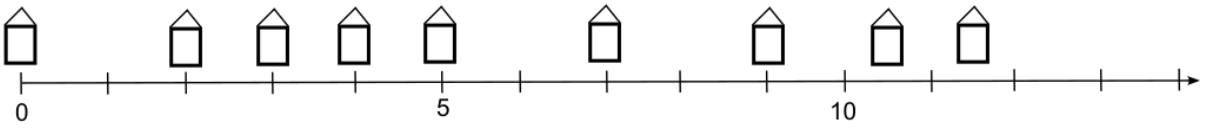
Éléments de correction sujet 7 (2024)

Exercice 1

1.
```
m1 = Maison(1)
m2 = Maison(3.5)
```
2.
```
a = Antenne(2.5, 1)
```
3.



4.
```
def creation_rue(pos):
    pos.sort()
    maisons = []
    for p in pos:
        m = Maison(p)
        maisons.append(m)
    return maisons
```

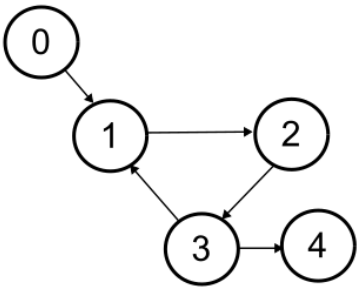
5.
```
def couvre(self, maison):
    return abs(self.get_pos_antenne - maison.get_pos_maison) <= self.get_rayon
```

6.
```
[0, 3, 7, 10.5]
```
7.

8.
```
def strategie_2(maisons, rayon):
    antennes = [Antenne(maisons[0].get_pos_maison() + rayon, rayon)]
    for m in maisons[1:]:
        if not antennes[-1].couvre(m):
            antennes.append(Antenne(m.get_pos_maison() + rayon, rayon))
    return antennes
```
9.

On garde la même structure (une simple boucle) dans les 2 stratégies, on a donc dans les 2 cas un coût linéaire en nombre de maisons.

Exercice 2

1. C’est un graphe orienté
2. réaliser la tâche (f) puis la tâche (g) : oui
   réaliser la tâche (g) puis la tâche (f) : non
   réaliser la tâche (i) puis la tâche (j) : oui (pas de dépendance)
   réaliser la tâche (j) puis la tâche (i) : oui (pas de dépendance)
3. il faut avoir réalisé les tâches : a, c, h, i et j
4. Le graphe ne contient pas de cycle
5. 0, 2 puis 1
   3, 5 puis 4
6. 

7. impossible, car on a la présence d’un cycle (pour réaliser 1 il faut déjà avoir fait 0 et 3, mais pour faire 3, il faut avoir fait 1)
8.

Appel mystere	variable ouverts	variable fermes
Avant l’appel mystere	[F,F,F,F,F]	[F,F,F,F,F]
mystere(M,1,5,[F,F,F,F,F], [F,F,F,F,F],None)	[F,T,F,F,F]	[F,F,F,F,F]
mystere(M,2,5,[F,T,F,F,F], [F,F,F,F,F],None)	[F,T,T,F,F]	[F,F,F,F,F]
mystere(M,3,5,[F,T,T,F,F], [F,F,F,F,F],None)	[F,T,T,T,F]	[F,F,F,F,F]
mystere(M,1,5,[F,T,T,T,F], [F,F,F,F,F],None)	[F,T,T,T,F]	[F,F,F,F,F]

   ok = False car au niveau du dernier appel récursif, ouverts[1] est déjà T, la fonction renvoie donc False
9. La fonction renvoie False quand le graphe comporte un cycle

10.
elt est égale à 2
11.
À la ligne 24 il faut écrire :
`resultat.empiler(s)`

Exercice 3

1.
```
personneA = Personne(112, ‘LESIEUR’, ‘Isabelle’,1982,2005)
```
2.
```
personneA.num_badge
```
3.
```
def annee_anciennete(self):
    return 2024 - self.annee_entree
```
4.
```
def ajouter(self, p):
    self.liste.append(p)
```
5.
```
def effectif(self):
    return len(self.liste)
```
6.
```
def donne_nom(self, num):
    for elt in self.liste:
        if elt.num_badge == num:
            return elt.nom
    return None
```
7.
```
def nb_personne_honneur(self, annee):
    nb_pers = 0
    for elt in self.liste:
        if annee - 10 == elt.annee_entree :
            nb_pers = nb_pers + 1
    return nb_pers
```
8.
```
def  plus_anciens(self):
    tab_ancien = []
    a_ancien = 100000
    for elt in self.liste:
        if elt.annee_entree == a_ancien :
            tab_ancien.append(elt.num_badge)
        if elt.annee_entree < a_ancien :
            a_ancien = elt.annee_entree
            tab_ancien = [elt.num_badge]
    return tab_ancien
```

9. Permet d’avoir le nom et le prénom de toutes les personnes qui travaillent dans le centre 2.
10.
```
UPDATE Personnel
SET num_centre = 3
WHERE num_badge = 135
```
11. Cela permet d’éviter la duplication d’informations
12. À l’aide d’une jointure entre l’attribut num de la table Centre et l’attribut num_centre (clé étrangère) de la table Personnel
13.
```
SELECT Personnel.nom
FROM Personnel
JOIN Centre ON num_centre = num
WHERE ville = “Lille” AND annee_debut >= 2015 AND annee_debut <= 2020
```
14. Certaines entrées de la table Personnel vont pointer vers le centre 1 alors qu’il n’existe plus, ce qui va poser un problème. Pour éviter ce problème, il faut réaffecter le personnel de ce centre avant de supprimer ce centre de la table Centre.

    Remarque : il n’ y a pas de * dans une requête de type DELETE