Proposition de correction

Exercice 1

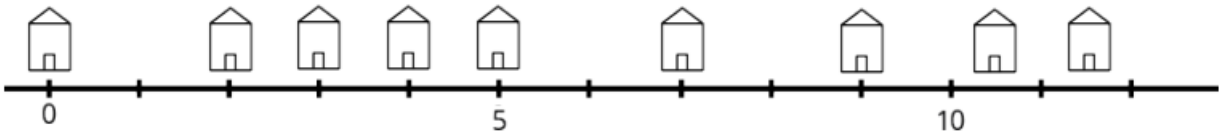
Q1

```
m1 = Maison(1)
m2 = Maison(3.5)
```

Q2

```
a = Antenne(2.5, 1)
```

Q3



Q4

```
def creation_rue(pos : list) -> list:
    """
    @param pos -- liste contenant des nombres correspondant aux positions des maisons.
    @return une liste d'objets de type Maison.
    """
    pos.sort()
    maisons = []
    for p in pos:
        m = Maison(p)
        maisons.append(m)
    return maisons
```

Q5

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

Q6

>>> maisons = creation_rue([0, 2, 3, 4, 5, 7, 9, 10.5, 11.5])

liste des maisons positionnées comme indiqué en Q3

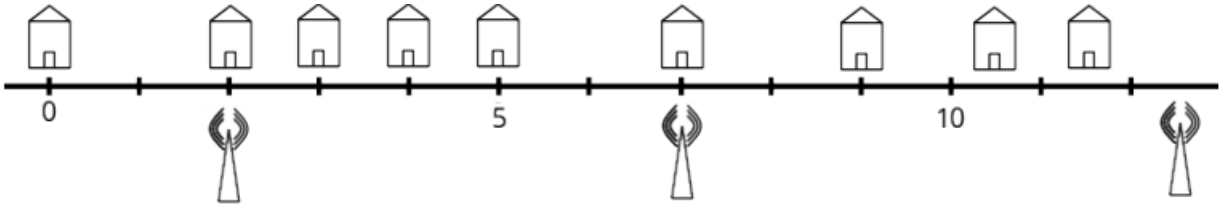
>>> antennes = strategie_1(maisons, 2)

liste des antennes qui couvrent les maisons

>>> print([a.get_pos_antenne() for a in antennes])

liste des positions des antennes qui couvrent les maisons

Q7



Q8

```
def strategie_2(maisons : list, rayon : float) -> list:
    """ place les antennes de façon optimum
    @param maisons -- une liste de maisons
    @param le rayon d'action des antennes
    @return une liste d'antennes
    @remark          la liste des maisons doit être triée par positions croissantes
    """
    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
```

Q9

- strategie_1 : O(n), coût linéaire
- strategie_2 : O(n), coût linéaire

Exercice 2

Q1

orienté

Q2

- 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
- réaliser la tâche (j) puis la tâche (i) : oui

Q3

a, c, h, i, j

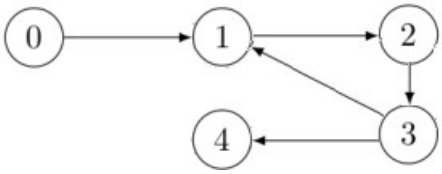
Q4

non

Q5

- 0, 2, 1
- 3, 5, 4

Q6



Q7

non, car le graphe est cyclique

Q8

False

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]

Q9

La fonction mystere implémente une forme de recherche en profondeur pour détecter des cycles dans un graphe à partir de sa matrice d'adjacence. Dans ce cas la fonction retourne False.

Q10

2

Q11

```
resultat.empiler(s)
```

Exercice 3

Partie A

Q1

```
personneA = Personne(112, 'LESIEUR', 'Isabelle', 1982, 2005)
```

Q2

```
print(personneA.num_badge)
```

Q3

```
def annee_anciennete(self) -> int:
    return 2024 – self.annee_entree
```

Q4

```
def ajouter(self, personne : Personne):
    self.liste.append(personne)
```

Q5

```
def effectif(self) -> int:
    return len(self.liste)
```

Q6

```
def donne_nom(self, num : int) -> str:
    """
    @param      num -- n° de badge
    @return     nom de la personne qui possède le n° badge, None sinon
    """
    for elt in self.liste:
        if elt.num_badge == num:
            return elt.nom
    return None
```

Q7

```
def nb_personne_honneur(self, annee : int) -> int:
    """
    @param      annee -- année de la cérémonie
    @return     le nombre de personne(s) à mettre à l'honneur.
    """
    anciennete = 10
    total      = 0
    for elt in self.liste:
        if elt.annee_anciennete() == anciennete:
            total += 1
    return total
```

Q8

```
def plus_anciens(self) -> list:
    """
    @return la liste des numéros de badge des personnes
        ayant la plus grande ancienneté dans l'entreprise.
    """
    liste   = []
    maximum = 0
    for elt in self.liste:
        if elt.annee_anciennete() >= maximum:
            if elt.annee_anciennete() > maximum:
                maximum = elt.annee_anciennete()
                liste   = []
            liste.append(elt.num_badge)
    return liste
```

Partie B

Q9

Sélectionne les nom et prénom des personnes affectées au centre n° 2

Q10

UPDATE Personnel

SET num_centre = 3

WHERE num_badge = 135

Q11

- Élimination des Redondances
- Amélioration de la Cohérence et de l'Intégrité des Données
- Facilité de Mise à Jour
- Optimisation des Performances

Q12

grâce à la clef étrangère num_centre de la table Personnel qui pointe sur la clef primaire num de la table Centre

Q13

SELECT Personnel.nom

FROM Personnel, Centre

WHERE Centre.nom = 'Lille'

AND Personnel.num_centre = Centre.num

AND Personnel.annee_debut BETWEEN 2015 AND 2020

ORDER BY Personnel.nom

Q14

La syntaxe correcte pour une requête DELETE n'inclut pas *

Pour supprimer toutes les colonnes dans une requête SELECT, voici la syntaxe correcte :

DELETE FROM Centre

WHERE nom = 'Normandie'

Attention : si la table Centre est référencée par la table Personnel via la clef étrangère num_centre, alors cela peut provoquer une erreur de contrainte référentielle.