Ce sujet, noté sur 30, comporte trois exercices notés respectivement sur 8, 10 et 12 points. Les trois exercices sont à traiter en intégralité. Ce sujet n'est pas à rendre avec votre copie. Lors de la correction, la plus grande attention sera portée à la clarté de vos réponses.

Le fait de répondre sans faire de phrases sera sévèrement sanctionné.

Lorsque vous écrirez du code, pensez à bien marquer de manière visible les indentations éventuelles.

Exercice 1 ***8 points***

L'énoncé de cet exercice utilise la convention suivante : les clés primaires seront soulignées et les clés étrangères seront précédées d'un #.

Le satellite GAIA a pour mission de cartographier un très grand nombre d'objets autour du Système Solaire. Régulièrement un catalogue est produit pour publier les données obtenues. Il est disponible sous différents formats dont par exemple sous forme de fichier `csv`.

`N_Obj` identifie chaque objet cartographié de manière unique.

Voici un extrait du catalogue :

N_Obj	N_Syst	Nom_Syst	Type	Nom_Obj	Asc_Droite	Decl	Parallaxe	Nom_SIMB
1	1	alf Cen	LM	Proxima Cen	217 392	-62 676	768 067	alf Cen C
2	1	alf Cen	Planet	Proxima Cen b	217 392	-62 676	768 067	
3	1	alf Cen	*	alf Cen A	219 902	-60 834	743 000	alf Cen A
4	1	alf Cen	*	alf Cen B	219 896	-60 838	743 000	alf Cen B
5	2	Barnard's Star	LM	Barnard's Star	269 449	4 739	546 976	Barnard's Star
6	3	Luhman 16	BD	Luhman 16 A	162 309	-53 318	501 557	Luhman 16A
7	3	Luhman 16	BD	Luhman 16 B	162 308	-53 318	501 557	Luhman 16B
9	5	Wolf 359	LM	Wolf 359	164 103	7 003	415 179	Wolf 359
10	6	HD 95735	LM	HD 95735	165 831	35 949	392 753	HD 95735
11	6	HD 95735	Planet	Lalande 21185 b	165 831	35 949	392 753	
12	7	alf CMa	*	alf CMa A	101 287	-16 716	379 210	alf CMa A
13	7	alf CMa	WD	alf CMa B	101 287	-16 721	374 490	alf CMa B
14	8	G 272-61	LM	G 272-61 A	24 772	-17 948	367 712	G 272-61A
15	8	G 272-61	LM	G 272-61 B	24 772	-17 948	373 844	G 272-61B
16	9	V1216 Sgr	LM	Ross 154	282 459	-23 837	336 027	Ross 154
17	10	HH And	LM	Ross 248	355 480	44 170	316 481	Ross 248

Pour manipuler plus facilement les données, un chercheur utilise un système de base de données relationnelle, dans lequel il crée le schéma relationnel de la table `Gaia` :

```
Gaia (N_Obj : Int, N_Syst : Int, Nom_Syst : String, #Type : String, Nom_Obj : String,
      Asc_Droite : Real, Decl : Real, Parallaxe : Real, Nom_SIMB : String)
```

Partie A : Schéma relationnel

1. Justifier pourquoi l'attribut `N_Obj` a été choisi comme clé primaire de la table `Gaia`.

Le type de l'objet (attribut `Type`) n'est pas une information directement compréhensible et le chercheur décide de créer une nouvelle table appelée `Typologie`. La clé primaire est `Type`.

Soit la table `Typologie` contenant les informations suivantes :

Type	Libelle_Type
LM	Etoile de faible masse
Planet	Planète
*	Etoile
BD	Naine Brune
WD	Naine Blanche

2. Proposer le schéma relationnel de la table `Typologie`.

3. Parmi les 4 commandes suivantes, une seule ne provoque pas d'erreur. Pour chacune de ces commandes, indiquer si elle provoque une erreur ou pas. En cas d'erreur, expliquer la cause de cette erreur.

a.
```
INSERT INTO Gaia VALUES ('8', 4, 'WISEA J085510', 'Naine Brune',
'WISEA J085510', 133.781, -7.244, 439.000, 'WISEA J085510') ;
```

b.
```
INSERT INTO Gaia VALUES (9, 4, 'WISEA J085510', 'Naine Brune',
'WISEA J085510', 133.781, -7.244, 439.000, 'WISEA J085510') ;
```

c.
```
INSERT INTO Gaia VALUES (8, 4, 'WISEA J085510', 'Naine Brune',
'WISEA J085510', 133.781, -7.244, 439.000, 'WISEA J085510') ;
```

d.
```
INSERT INTO Gaia VALUES (8, 4, 'WISEA J085510', 'Naine Brune',
'WISEA J085510', '133.781', -7.244, 439.000, 'WISEA J085510') ;
```

4. Expliquer pourquoi le code `SQL` suivant ne fonctionne pas :

```
INSERT INTO Typologie VALUES ('BD', 'Trou Noir') ;
```

5. Indiquer le résultat de la requête suivante exécutée sur l'extrait présenté :

```
SELECT N_Obj, Parallaxe FROM Gaia WHERE Type = 'Planet' ;
```

6. Écrire la requête qui permet de récupérer le nom du système, le nom de l'objet et le libellé du type pour des objets ayant une parallaxe supérieure à 400 000 et étant des 'Etoile de faible masse'. La requête proposée utilisera obligatoirement une jointure.

Exercice 2 *10 points*

On s'intéresse dans cet exercice à un algorithme de mélange des éléments d'une liste.

1. Pour la suite, il sera utile de disposer d'une fonction `echange` qui permet d'échanger dans une liste `lst` les éléments d'indice `i1` et `i2`. Expliquer pourquoi le code Python ci-dessous ne réalise pas cet échange et en proposer une modification.

```
def echange(lst, i1, i2):
    lst[i2] = lst[i1]
    lst[i1] = lst[i2]
```

2. La documentation du module `random` de Python fournit les informations ci- dessous concernant la fonction `randint(a,b)` :

`Renvoie un entier aléatoire N tel que a <= N <= b. Alias pour randrange(a,b+1).`

Parmi les valeurs ci-dessous, quelles sont celles qui peuvent être renvoyées par l'appel `randint(0,10)` ?

`0` `1` `3.5` `9` `10` `11`

3. Le mélange de Fischer Yates est un algorithme permettant de permuter aléatoirement les éléments d'une liste. On donne ci-dessous une mise en œuvre récursive de cet algorithme en Python.

```
from random import randint
def melange(lst, ind):
    print(lst)
    if ind > 0:
        j = randint(0, ind)
        echange(lst, ind, j)
        melange(lst, ind-1)
```

a. Expliquer pourquoi la fonction `melange` se termine toujours.

b. Lors de l'appel de la fonction `melange`, la valeur du paramètre `ind` doit être égal au plus grand indice possible de la liste `lst`. Pour une liste de longueur n, quel est le nombre d'appels récursifs de la fonction `melange` effectués, sans compter l'appel initial ?

c. On considère le script ci-dessous :

```
lst = [v for v in range(5)]
melange(lst, 4)
```

On supppose que les valeurs successivement renvoyées par la fonction `randint` sont `2`, `1`, `2` et `0`.
Les deux premiers affichages produits par l'instruction `print(lst)` de la fonction `melange` sont :

```
[0, 1, 2, 3, 4]
[0, 1, 4, 3, 2]
```

Donner les affichages suivants produits par la fonction `melange`.

d. Proposer une version itérative du mélange de Fischer Yates.

Exercice 3 *12 points*

Dans un entrepôt de e-commerce, un robot mobile autonome exécute successivement les tâches qu'il reçoit tout au long de la journée.

La mémorisation et la gestion de ces tâches sont assurées par une structure de données.

1. Dans l'hypothèse où les tâches devraient être extraites de cette structure (pour être exécutées) dans le même ordre que leur ordre d'arrivée dans la structure, préciser si ce fonctionnement traduit le comportement d'une file ou d'une pile. Justifier.

En réalité, selon l'urgence des tâches à effectuer, on associe à chacune d'elles, lors de la mémorisation, un indice de priorité (nombre entier) distinct : il n'y a pas de valeur en double.

Plus cet indice est faible, plus la tâche doit être traitée prioritairement.

La structure de données retenue est assimilée à un arbre binaire de recherche (ABR) dans lequel chaque nœud correspond à une tâche caractérisée par son indice de priorité.

Rappel : Dans un arbre binaire de recherche, chaque nœud est caractérisé par une valeur (ici l'indice de priorité), telle que chaque nœud du sous-arbre gauche a une valeur strictement inférieure à celle du nœud considéré, et que chaque nœud du sous-arbre droit possède une valeur strictement supérieure à celle-ci.
Cette structure de données présente l'avantage de mettre efficacement en œuvre l'insertion ou la suppression de nœuds, ainsi que la recherche d'une valeur.

Par exemple, le robot a reçu successivement, dans l'ordre, des tâches d'indice de priorité 12, 6, 10, 14, 8 et 13. En partant d'un arbre binaire de recherche vide, l'insertion des différentes priorités dans cet arbre donne la Figure 1.

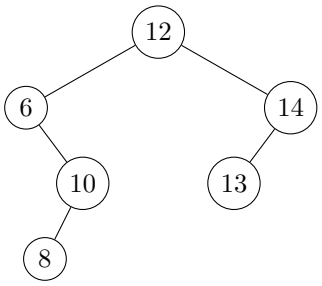

Figure 1 – Exemple d'un arbre binaire

2. En utilisant le vocabulaire couramment utilisé pour les arbres, par quel mot désigne-t-on le nombre de tâches restant à effectuer, c'est-à-dire le nombre total de nœuds de l'arbre ?

3. Lorsque le robot reçoit une nouvelle tâche, on déclare un nouvel objet, instance de la classe `Noeud`, puis on l'insère dans l'arbre binaire de recherche (instance de la classe `ABR`) du robot. Ces 2 classes sont définies comme suit :

```
1  class Noeud:
2      def __init__(self, tache, indice):
3          self.tache = tache       # ce que doit accomplir le robot
4          self.indice = indice     # indice de priorité (int)
5          self.gauche = ABR()      # sous-arbre gauche vide (ABR)
6          self.droite = ABR()      # sous-arbre droit vide (ABR)

7  class ABR:
8      """ arbre binaire de recherche initialement vide """
9      def __init__(self):
10         self.racine = None # arbre vide
11         #Remarque : si l'arbre n'est pas vide, racine est une instance de la classe Noeud

12     def est_vide(self):
13         """ renvoie True si l'arbre autoréférencé est vide, False sinon """
14         return self.racine == None

15     def insere(self, nouveau_noeud):
16         """ insère un nouveau noeud, instance de la classe Noeud, dans l'ABR """
17         if self.est_vide():
18             self.racine = nouveau_noeud
19         elif self.racine.indice ... nouveau_noeud.indice:
20             self.racine.gauche.insere(nouveau_noeud)
21         else :
22             self.racine.droite.insere(nouveau_noeud)
```

a. Donner les noms des attributs de la classe `Noeud`.

b. Expliquer en quoi la méthode `insere` est dite récursive et justifier rapidement qu'elle se termine.

c. Indiquer le symbole de comparaison manquant dans le test à la **ligne 19** de la méthode `insere` pour que l'arbre binaire de recherche réponde bien à la définition de l'encadré « Rappel » de la page précédente.

d. On considère le robot dont la liste des tâches est représentée par l'arbre de la FIGURE 1. Ce robot reçoit, successivement et dans l'ordre, des tâches d'indice de priorité 11, 5, 16 et 7, sans avoir accompli la moindre tâche entretemps. Recopier et compléter la FIGURE 1 après l'insertion de ces nouvelles tâches.

4. Avant d'insérer une nouvelle tâche dans l'arbre binaire de recherche, il faut s'assurer que son indice de priorité n'est pas déjà présent.
Écrire une méthode `est_present` de la classe `ABR` qui répond à la description :

```
def est_present(self, indice_recherche) :
    """renvoie True si l'indice de priorité indice_recherche (int) passé en paramètre est déjà
    ↪  l'indice d'un noeud de l'arbre, False sinon"""
```

5. Comme le robot doit toujours traiter la tâche dont l'indice de priorité est le plus petit, on envisage un parcours infixe de l'arbre binaire de recherche.

a. Donner l'ordre des indices de priorité obtenus à l'aide d'un parcours infixe de l'arbre binaire de recherche de la FIGURE 1.

b. Expliquer comment exploiter ce parcours pour déterminer la tâche prioritaire.

6. Afin de ne pas parcourir tout l'arbre, il est plus efficace de rechercher la tâche du nœud situé le plus à gauche de l'arbre binaire de recherche : il correspond à la tâche prioritaire.
Recopier et compléter la méthode récursive `tache_prioritaire` de la classe `ABR` :

```
def tache_prioritaire(self):
    """renvoie la tache du noeud situé le plus à gauche de l'ABR supposé non vide"""
    if self.racine..........est_vide():#pas de noeud plus à gauche
        return self.racine.........
    else:
        return self.racine.gauche.........()
```