EXERCICE 3 (4 points)

Cet exercice est consacré aux arbres binaires de recherche et à la notion d'objet.

1. Voici la définition d'une classe nommée `ArbreBinaire`, en Python :

Numéro de lignes	Classe `ArbreBinaire`
1	`class ArbreBinaire:`
2	`    """ Construit un arbre binaire """`
3	`    def __init__(self, valeur):`
4	`        """ Crée une instance correspondant`
5	`        à un état initial """`
6	`        self.valeur = valeur`
7	`        self.enfant_gauche = None`
8	`        self.enfant_droit = None`
9	`    def insert_gauche(self, valeur):`
10	`        """ Insère le paramètre valeur`
11	`        comme fils gauche """`
12	`        if self.enfant_gauche is None:`
13	`            self.enfant_gauche = ArbreBinaire(valeur)`
14	`        else:`
15	`            new_node = ArbreBinaire(valeur)`
16	`            new_node.enfant_gauche = self.enfant_gauche`
17	`            self.enfant_gauche = new_node`
18	`    def insert_droit(self, valeur):`
19	`        """ Insère le paramètre valeur`
20	`        comme fils droit """`
21	`        if self.enfant_droit is None:`
22	`            self.enfant_droit = ArbreBinaire(valeur)`
23	`        else:`
24	`            new_node = ArbreBinaire(valeur)`
25	`            new_node.enfant_droit = self.enfant_droit`
26	`            self.enfant_droit = new_node`
27	`    def get_valeur(self):`
28	`        """ Renvoie la valeur de la racine """`
29	`        return self.valeur`
30	`    def get_gauche(self):`
31	`        """ Renvoie le sous arbre gauche """`
32	`        return self.enfant_gauche`
33	`    def get_droit(self):`
34	`        """ Renvoie le sous arbre droit """`
35	`        return self.enfant_droit`

a. En utilisant la classe définie ci-dessus, donner un exemple d'attribut, puis un exemple de méthode.

b. Après avoir défini la classe `ArbreBinaire`, on exécute les instructions Python suivantes :

```
r = ArbreBinaire(15)
r.insert_gauche(6)
r.insert_droit(18)
a = r.get_valeur()
b = r.get_gauche()
c = b.get_valeur()
```

Donner les valeurs associées aux variables `a` et `c` après l'exécution de ce code.

On utilise maintenant la classe `ArbreBinaire` pour implémenter un arbre binaire de recherche.

On utilisera la définition suivante : un arbre binaire de recherche est un arbre binaire, dans lequel :

- on peut comparer les valeurs des nœuds : ce sont par exemple des nombres entiers, ou des lettres de l'alphabet.
- si `x` est un nœud de cet arbre et `y` est un nœud du sous-arbre gauche de `x`, alors il faut que `y.valeur <= x.valeur`.
- si `x` est un nœud de cet arbre et `y` est un nœud du sous-arbre droit de `x`, alors il faut que `y.valeur => x.valeur`.

2. On exécute le code Python suivant. Représenter graphiquement l'arbre ainsi obtenu.

```
racine_r = ArbreBinaire(15)

racine_r.insert_gauche(6)
racine_r.insert_droit(18)

r_6 = racine_r.get_gauche()
r_6.insert_gauche(3)
r_6.insert_droit(7)

r_18 = racine_r.get_droit()
r_18.insert_gauche(17)
r_18.insert_droit(20)

r_3 = r_6.get_gauche()
r_3.insert_gauche(2)
```

3. On a représenté sur la figure 1 ci-dessous un arbre. Justifier qu'il ne s'agit pas d'un arbre binaire de recherche. Redessiner cet arbre sur votre copie en conservant l'ensemble des valeurs {2,3,5,10,11,12,13} pour les nœuds afin qu'il devienne un arbre binaire de recherche.

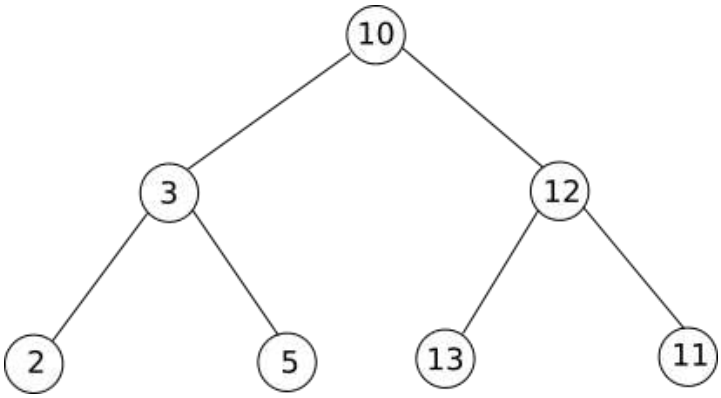

Figure 1

4. On considère qu'on a implémenté un objet `ArbreBinaire` nommé A représenté sur la figure 2.

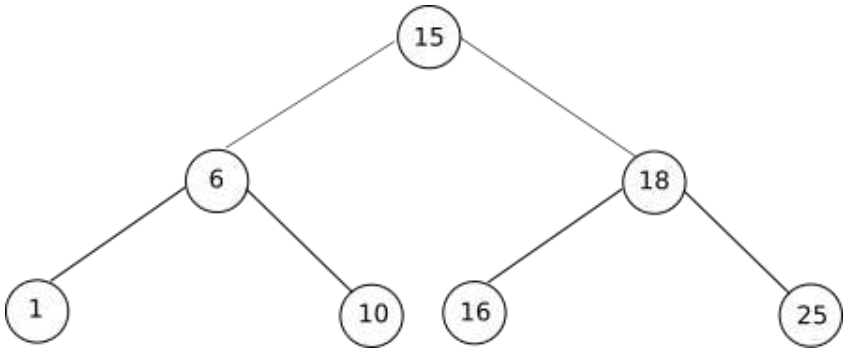

Figure 2

On définit la fonction `parcours_infixe` suivante, qui prend en paramètre un objet `ArbreBinaire T` et un second paramètre `parcours` de type liste.

Numéro de lignes	Fonction `parcours_infixe`
1	`def parcours_infixe(T, parcours):`
2	`    """ Affiche la liste des valeurs de l'arbre """`
3	`    if T is not None:`
4	`        parcours_infixe(T.get_gauche(), parcours)`
5	`        parcours.append(T.get_valeur())`
6	`        parcours_infixe(T.get_droit(), parcours)`
7	`    return parcours`

Donner la liste renvoyée par l'appel suivant : `parcours_infixe(A,[ ])`.