TD: implémentation et algorithmes avec les arbres binaires de recherche:

Exercice 1: Arbre binaire de recherche:

1. vérifier que l'arbre de la figure 1 est un arbre binaire de recherche:

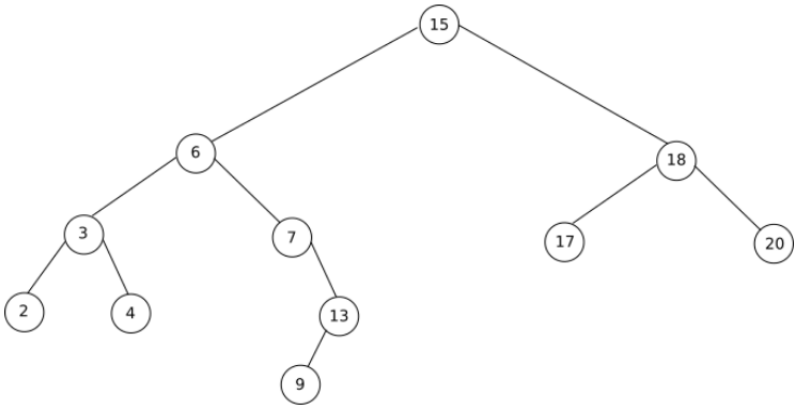

Figure 1: Arbre binaire de recherche

2. Dessiner un arbre binaire de recherche contenant les valeurs 11, 13, 14, 15, 17, 18, 19 satisfaisant à la condition donnée dans chaque cas:

   2.1 la hauteur de l'arbre est 3.

   2.2 ressemblant à l'arbre suivant:

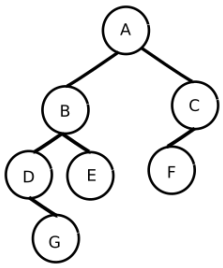

Figure 2: Arbre binaire de recherche

Exercice 2: Arbre binaire de recherche (fonctions): On s'interesse au programme suivant:

```python
def creer_arbre(r = None):
    """renvoie un arbre vide ou un arbre de racine r"""
    if r:
        return [r, [],[]]
    else:
        return []

def arbre_vide(a):
    return a == []


def fils_gauche(a):
    if not arbre_vide(a):
        return a[1]

def fils_droit(a):
    if not arbre_vide(a):
        return a[2]

def insere(a,val):
    """fonction recursive"""
    if arbre_vide(a):
        a.append(val)
        a.append([])
        a.append([])
    elif val <= a[0]:
        insere(a[1],val)
    else:
        insere(a[2],val)
```

arbre_binaire_relation_ordre.py

1. Expliquer avec précision le rôle de chacune de ces fonctions. En quoi la fonction insere() est-elle récursive?
2. En quoi l'arbre de la figure 1 est-il un arbre binaire de recherche?
3. A l'aide de l'éditeur Python, implémenter l'arbre de la figure 3 et le représenter.
4. Ecrire une fonction *recherche* permettant de trouver une clé dans cet arbre
5. Ecrire une fonction itérative *minimum_it* permettant de trouver la clé minimale dans cet arbre.
6. Ecrire une fonction récursive *minimum_rec* permettant de trouver la clé minimale dans cet arbre.

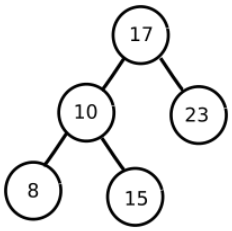

Figure 3: Arbre binaire de recherche

Exercice 3: Arbre binaire de recherche (classe v1): On s'interesse au programme suivant:

```python
class Noeud:
    """Arbre binaire de recherche implemente par ses noeuds"""
    def __init__(self, valeur):
        self.valeur = valeur
        self.parent = None
        self.gauche = None
        self.droite = None

    def __str__(self):
        return str(self.valeur)

    def ajoute(self,valeur):
        if valeur < self.valeur:
            if self.gauche is None:
                self.gauche = Noeud(valeur)
                self.gauche.parent = self
            else:
                self.gauche.ajoute(valeur)
        elif valeur > self.valeur:
            if self.droite is None:
                self.droite = Noeud(valeur)
                self.droite.parent = self
            else:
                self.droite.ajoute(valeur)
```

arbre_binaire_recherche1.py

1. Expliquer avec précision le rôle de chacune des méthodes de la classe Noeud. Une des fonctions est-elle récursive?
2. A l'aide de l'éditeur Python, implémenter l'arbre de la figure 3 et le représenter.

Exercice 4: Arbre binaire de recherche (classe v2): On s'interesse au programme suivant:

```python
class Arbre:
    def __init__(self,val):
        self.valeur = val
        self.gauche = None
        self.droit = None

    def insere(self,val):
        if val < self.valeur:
            if self.gauche != None:
                self.gauche.insere(val)
            else:
                self.gauche = Arbre(val)
        else:
            if self.droit != None:
                self.droit.insere(val)
            else:
                self.droit = Arbre(val)
```

arbre_binaire_recherche2.py

1. Expliquer avec précision le rôle de chacune des méthodes de la classe Noeud. Une des fonctions est-elle récursive?
2. A l'aide de l'éditeur Python, implémenter l'arbre de la figure 3 et le représenter.

Exercice 4: Recherche d'une clé: Pour la classe Noeud (v1) ou pour la classe Arbre (v2), ajouter une méthode *recherche(val)* qui retournera Oui si la valeur val est dans l'arbre binaire de recherche et Non sinon.

Exercice 5: Recherche d'un maximum ou d'un minimum: Dans la classe Noeud (v1) ou pour la classe Arbre (v2), ajouter une méthode *maximum()* et *minimum()* qui retournera le maximum et le minimum de l'arbre binaire de recherche.