IMPORTANT. Durée **3 heures**. Il sera tenu compte de la **clarté** et de la **qualité** de votre copie. Aucun document autorisé.

1 Listes chaînées

Remarque : *Dans la suite, que ce soit pour les listes chaînées ou pour les arbres, on classe les méthodes par difficulté, il s'agit de difficulté liée à la compréhension, pas une mesure de la longueur du code.*

On considère les classes suivantes :

```
public class Noeud {

    private int val;
    private Noeud suivant;

    public Noeud(int x, Noeud n) {
        this.val = x;
        this.suivant = n;
    }
}
```

```
public class Liste {

    private Noeud tete;

    public Liste() {
        this.tete = null;
    }

    public void ajouter(int x) {
        this.tete =
            new Noeud(x, this.tete);
    }
}
```

On représentera les listes par des suites d'entiers.
Par exemple, $(1; 4; 5)$ représentera une liste de 3 éléments dont le premier Noeud contient 1, le

lui-même compris. Par exemple, si la liste avant application de la méthode est $(4;1;2;5)$ la liste après application est $(12;8;7;5)$, où par exemple 8 est bien la somme de 1, 2 et 5.

Indication : on utilisera le fait que la nouvelle valeur d'un élément est son ancienne valeur + la somme de l'élément suivant.

2 Arbres

On considère les classes suivantes :

```
public class Arbre {

    private NoeudArbre racine;

    public Arbre () {
        this.racine = null;
    }
}
```

```
public class NoeudArbre {

    private int etiquette;
    private NoeudArbre gauche;
    private NoeudArbre droit;

    public NoeudArbre(int etiquette,
                      NoeudArbre gauche,
                      NoeudArbre droit) {
        this.etiquette = etiquette;
        this.gauche = gauche;
        this.droit = droit;
    }

    public NoeudArbre(int etiquette) {
        this.etiquette = etiquette;
        this.gauche = null;
        this.droit = null;
    }
}
```

2.1 Questions

Les méthodes suivantes doivent êtres écrites dans `Arbre`, certaines de ces méthodes nécessiteront de coder des méthodes auxilliaires dans `NoeudArbre` ou dans `Arbre`. Vous devez donc **impérativement** préciser dans quelle classe est codée chaque méthode.

1. (facile) Écrivez une méthode `int etiquetteDroite()` qui retourne la valeur du nœud la plus à droite, c'est-à-dire celui qu'on obtient en allant à droite depuis la racine, le plus de fois possible. Si l'arbre est vide, on retournera -1. Par exemple, la méthode appliquée à l'arbre de la figure 1 donnera 1.
2. (plus difficile) Écrivez une méthode `void ajouteFeuille(int n)` qui à chaque feuille ajoute une feuille comme fils gauche et une feuille comme fils droit

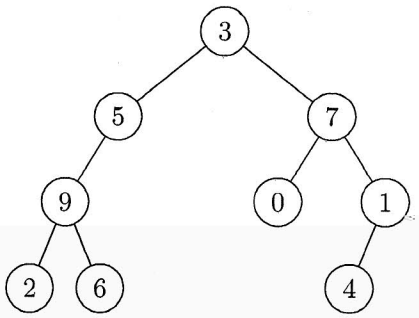

FIGURE 1 –

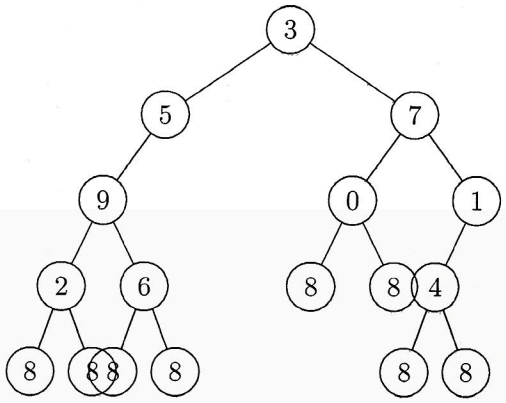

FIGURE 2 –

3 Exercice sur les objets

On s'intéresse à la modélisation d'un parc d'imprimantes. Chaque imprimante possède une marque, et un numéro de série. Elles sont chargées avec une certaine quantité de feuilles de papier, dont le nombre est inférieur à une limite propre à l'imprimante. Ces imprimantes contiennent un emplacement destiné à recevoir une cartouche d'encre. Ces cartouches sont elles mêmes des objets complexes : elles ont une contenance, exprimée en millilitres (un entier), une valeur indiquant le [illegible] (un pourcentage, c'est un entier entre 0 et 100), un indicateur qui précise [illegible] et une description de la couleur [illegible]

[illegible]