



UNIVERSITÉ DE MONTPELLIER

ANNALE D'EXAMENS

---

L2 Informatique

Semestre 2

---



Année Universitaire 2020 - 2021

## Introduction

Le présent document que vous lisez actuellement est un recueil d'annales de différentes matières concernant le parcours Informatique. Il n'est pas exhaustif et comporte quelques corrections officielles données par des enseignants du Département Informatique et du Département Mathématique de l'Université de Montpellier.

La base de données des annales de l'association **Hello World** est soumise à la bonne volonté d'anciens étudiants désirant partager leurs ressources personnelles et d'apporter leur soutien aux étudiants en Informatique. Une fois les périodes d'examens terminées, vous pouvez nous envoyer par mail les sujets scannés au format pdf ou jpg/png (pas de photo par téléphone à cause de la lisibilité).

Si vous souhaitez soutenir l'association dans ses actions, vous avez la possibilité d'adhérer à l'association ou faire un don afin que nous puissions continuer d'apporter de l'aide aux étudiants, de créer et soutenir des projets étudiants ainsi que d'organiser des événements culturels et de divertissement. Pour ce faire, c'est par ici : <http://bit.ly/AdhesionHelloWorld>

Vous pouvez nous retrouver sur notre page Facebook [@UMHelloWorld](#), sur le [Discord Asso Hello World](#) (<http://bit.ly/UMHelloWorld>) et à notre bureau au Bâtiment 38, le (S)pace à l'étage, bureau n°5 et boîte postale n°9 au sein de la Faculté des Sciences.

Vous pouvez également nous contacter par mail sur [umhelloworld@gmail.com](mailto:umhelloworld@gmail.com)

Bonnes révisions et bon courage pour vos examens finaux !  
L'équipe Hello World vous souhaite à tous de bonnes fêtes de fin d'année ! :D

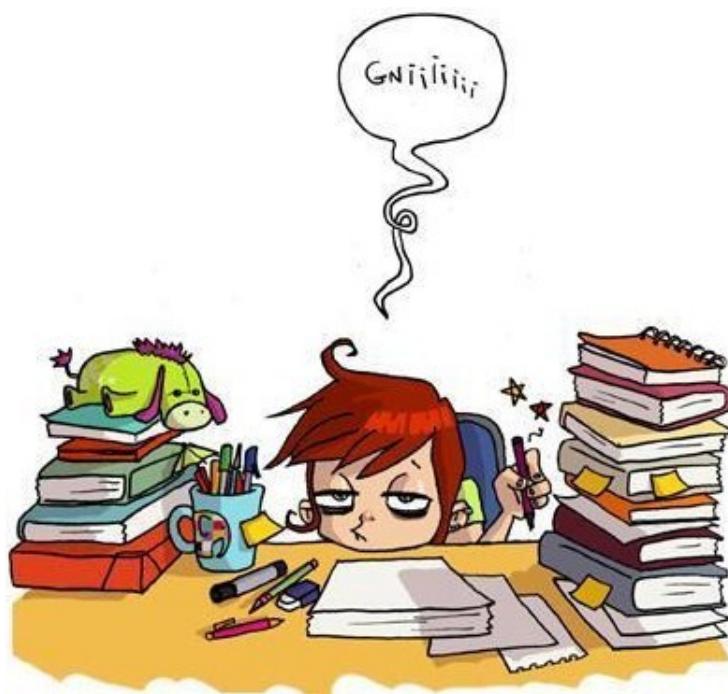


FIGURE 1 – frenchiesinlex.wordpress.com

# Examen de Logique 1 – HLIN402 – session 1

Michel Leclère

2 mai 2018

Durée : 2h. 1 feuille A4 manuscrite recto-verso autorisée. Pas de calculatrice.

**Question 1 (3 points)** Soit le connecteur *nor* défini par  $a \text{ nor } b \equiv \neg(a \vee b)$ . Montrez par induction sur le nombre  $n$  de connecteurs que toute formule bien formée de la logique des propositions est équivalente à une formule ayant *nor* comme seul connecteur. Dans cet exercice, il sera porté une grande attention à la qualité de la rédaction de votre preuve !

**Question 2 (2 points)** Modélisez (en expliquant vos choix) en logique des propositions les 5 phrases suivantes :

1. *Quand la musique est rythmée, il danse.*
2. *La musique rythmée ne le fait pas danser.*
3. *Il ne suffit pas que la musique soit rythmée pour qu'il danse.*
4. *Il n'a pas besoin de musique rythmée pour danser.*

**Question 3 (3 points)** Dans le Marchand de Venise, de Shakespeare, Portia a trois coffrets, un d'or, un d'argent et un de plomb et, dans l'un d'eux, elle a caché son portrait. Quand un des soupirants se présente, elle lui fait choisir l'un des coffrets, et c'est celui qui saura trouver le coffret contenant son portrait qui pourra l'épouser. Mais le couvercle de chaque coffret porte deux inscriptions pour guider le choix du soupirant, car Portia ne veut pas choisir un époux pour sa vertu mais pour son intelligence :

- Coffret en or :
  - Le portrait n'est pas dans ce coffret
  - Le portrait est dans le coffret en argent
- Coffret en argent :
  - Le portrait n'est pas dans le coffret en or
  - Le portrait est dans le coffret en plomb
- Coffret en plomb :
  - Le portrait n'est pas dans ce coffret
  - Le portrait est dans le coffret en or

Elle expliqua à son soupirant que sur l'un des coffrets deux affirmations étaient vraies, sur un autre elles étaient fausses toutes les deux, et sur le troisième l'une était vraie et l'autre était fausse. Quel coffret le candidat au mariage devrait-il choisir pour épouser Portia ?

1. Modélisez en logique des propositions ce problème. On choisira comme seuls symboles propositionnels :  $O = \text{"le portrait est dans le coffret en or"}$ ,  $A = \text{"le portrait est dans le coffret en argent"}$ ,  $P = \text{"le portrait est dans le coffret en plomb"}$ . On pourra par exemple modéliser le cas où c'est le coffret en or qui a une inscription vraie et une fausse et les deux autres qui sont pour l'une vraies et
2. Résolvez-le par la méthode de votre choix.

**Question 4 (4 points)** Montrez que la formule suivante est insatisfiable :

$$\neg(((r \vee t) \rightarrow (r \vee s)) \rightarrow (r \vee (t \rightarrow s)))$$

1. En utilisant la méthode des tableaux séquentiels. Vous expliquerez en quoi le tableau construit permet de conclure à l'insatisfiabilité de la formule.

2. En utilisant le système LK des séquents de Gentzen.

**Question 5 (2 points)** Dites si la formule suivante est satisfiable ou non en utilisant la méthode de résolution :

$$(a \leftrightarrow b \vee \neg(b \wedge a))$$

Vous séparerez bien mise sous forme clausale, simplification de l'ensemble de clauses obtenues, puis résolution et expliquerez pourquoi la résolution vous permet de conclure.

**Question 6 (3 points)** On s'intéresse à la *L-résolution*, une stratégie de résolution qui consiste à : (1) toujours réutiliser la dernière clause obtenue et (2) en effectuant une résolution avec une autre clause selon le symbole du premier littéral de cette clause (on voit donc les clauses comme des ensembles ordonnés). Par exemple si la dernière clause est  $\{\neg p, \neg q\}$ , on cherche à effectuer une résolution avec une clause contenant le littéral  $p$ . On suppose par ailleurs que l'on est dans un contexte dit de système d'interrogation où l'on dispose d'un ensemble de clauses définies  $D_1 \dots D_k$ , des clauses ayant exactement un littéral positif, et d'une unique clause requête  $R$ , une clause ne contenant que des littéraux négatifs. Exemple :  $D_1 = \{a, \neg c, \neg d\}$ ,  $D_2 = \{e, \neg b\}$ ,  $D_3 = \{f, \neg a, \neg e\}$ ,  $D_4 = \{a, \neg c, \neg e\}$ ,  $D_5 = \{b\}$ ,  $D_6 = \{c\}$  et  $R = \{\neg e, \neg f\}$ .

1. Montrez par récurrence sur la longueur d'une dérivation que, dans un tel contexte, la *L-résolution* ne produit que des clauses contenant uniquement des littéraux négatifs, quand la clause de départ ne contient que des littéraux négatifs.
2. Sur l'exemple précédent, donnez deux dérivations différentes obtenues selon cette stratégie, l'une conduisant à un arrêt par absence de *L-résolution* possible et l'autre à la clause vide.
3. Cette stratégie de résolution est-elle complète et pour quelles conditions de mise en œuvre ? Justifiez votre réponse.

**Question 7 (3 points)** Soit  $\mathcal{E}$  un ensemble de formules bien formées de la logique des propositions, dites si les propriétés suivantes sont correctes ou non :

Propriété 1 :  $\mathcal{E}$  est contradictoire si et seulement s'il existe une formule  $F \in \mathcal{E}$  telle que  $\mathcal{E} \setminus \{F\} \models \neg F$   
(c'est à dire  $\mathcal{E}$  privée de  $F$  a pour conséquence logique  $\neg F$ )

Propriété 2 :  $\mathcal{E}$  est consistant si et seulement s'il existe une formule  $F \in \mathcal{E}$  telle que  $\mathcal{E} \setminus \{F\} \not\models F$ .  
Vous justifierez vos réponses en donnant pour chacune des propriétés une preuve ou un contre-exemple pour chacun des sens de l'équivalence.

# Examen de Logique 1 – GLIN402 – session 1

Michel Leclère

14 Mai 2014

Durée : 2h. Tout document autorisé. Pas de calculatrice, portable, tablette...

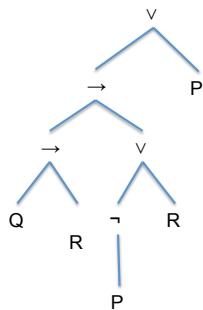
La note tiendra le plus grand compte de la manière dont vous aurez rédigé vos démonstrations et vos explications.

**Question 1 (2 points)** Soit la formule :

$$(((Q \rightarrow R) \rightarrow \neg P \vee R) \vee P)$$

a. Dessinez l'arborescence de cette formule.

arborescence ci-dessous



b. Dites si elle est valide, contingente ou insatisfiable en justifiant votre réponse.

valide

**Question 2 (3 points)** Soit le problème suivant :

H1 : Si Alice et Julie viennent à Montpellier, Zoé viendra aussi.

H2 : Si Alice ne vient pas à Montpellier, Julie non plus.

H3 : Julie ou Zoé, l'une des deux au moins, viendra à Montpellier.

Peut-on savoir qui viendra ou ne viendra pas à coup sûr à Montpellier ?

Vous modéliserez chacune des phrases en logique des propositions, expliquerez quels problèmes de logique des propositions permettent de résoudre cette question, et justifiez votre réponse.

H1 =  $A \wedge J \rightarrow Z$  ou une forme sémantiquement équivalente.

H2 =  $\neg A \rightarrow \neg J$  ou une forme sémantiquement équivalente.

H3 =  $J \vee Z$  ou une forme sémantiquement équivalente.

il s'agit d'étudier si  $A, \neg A, J, \neg J, Z, \neg Z$  sont des conséquences logiques de  $\{H1, H2, H3\}$ .

Zoé viendra et on ne sait pas pour les autres.

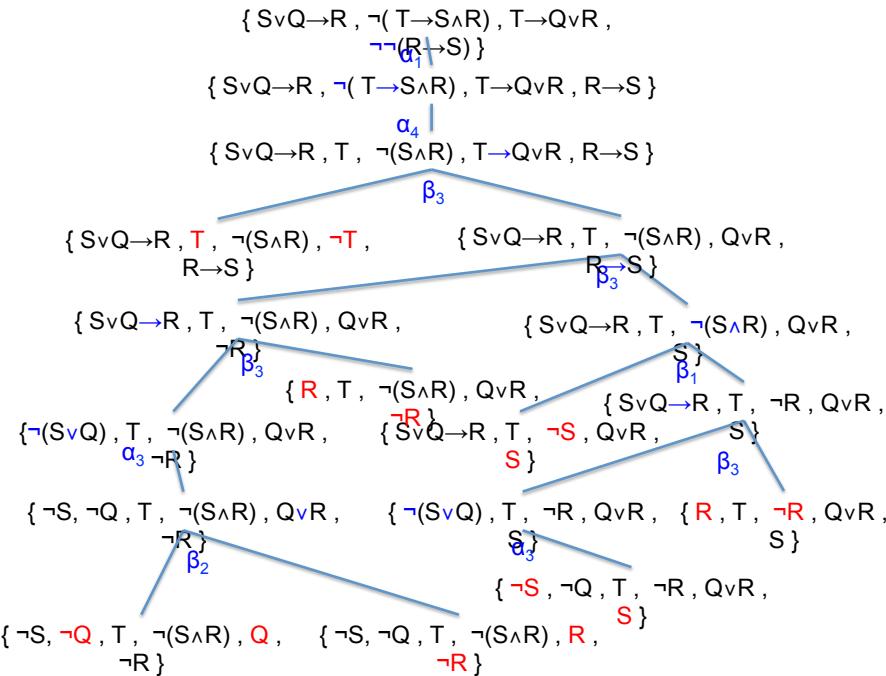
justification par table de vérité, résolution, etc.

**Question 3 (3 points)** En utilisant la méthode des tableaux séquentiels montrez la conséquence logique suivante (après avoir développé votre tableau, vous rappellerez quelles propriétés et/ou théorèmes permettent de conclure à la conséquence logique à partir de votre tableau) :

$$\{ S \vee Q \rightarrow R, \neg(T \rightarrow S \wedge R), T \rightarrow Q \vee R \} \models \neg(R \rightarrow S)$$

développement d'un tableau correct (cf. exemple ci-dessous).

toutes les feuilles fermées et on a fait un tableau des hypothèses et de la négation de la conclusion.



**Question 4 (4 points)** Soit les formules suivantes :

$$\mathcal{H}_1 = (D \vee \neg((C \rightarrow B) \rightarrow C)) \rightarrow A$$

$$\mathcal{H}_2 = (A \rightarrow C) \vee (A \wedge E) \vee (B \wedge E)$$

$$\mathcal{H}_3 = (C \vee E) \rightarrow (D \wedge (C \rightarrow E))$$

$$\mathcal{C} = A \wedge D \wedge E$$

Montrez à l'aide de la méthode de résolution la conséquence logique :  $\{\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\} \models \mathcal{C}$ . Vous détaillerez :

1. La mise sous forme clausale. Vous indiquerez alors quelles clauses peuvent être supprimées et par quelles propriétés.
2. La séquence de résolutions.
3. Les propriétés et/ou théorèmes qui permettent de conclure à la conséquence logique à partir de votre séquence de résolution.

mise sous forme clausale de chaque élément

$$FC(\mathcal{H}_1) = \{ \{A, \neg D\}, \{A, C\}, \{A, \neg B, C\} \}$$

$$FC(\mathcal{H}_2) = \{ \{A, \neg A, B, C\}, \{\neg A, C, E\} \}$$

$$FC(\mathcal{H}_3) = \{ \{\neg C, D\}, \{\neg C, E\}, \{D, \neg E\}, \{\neg C, E, \neg E\} \}$$

$$FC(\neg C) = \{ \{\neg A, \neg D, \neg E\} \}$$

simplifications de la FC : la première clause de  $FC(\mathcal{H}_2)$  et la dernière de  $FC(\mathcal{H}_3)$  peuvent être supprimées car tautologiques ; la troisième clause de  $FC(\mathcal{H}_1)$  peut être supprimée car subsumée par la deuxième clause de  $FC(\mathcal{H}_1)$ . Au final on a :

$$FC = \{ C_1 = \{A, \neg D\}, C_2 = \{A, C\}, C_3 = \{\neg A, C, E\}, C_4 = \{\neg C, D\}, C_5 = \{\neg C, E\}, C_6 = \{D, \neg E\}, C_7 = \{\neg A, \neg D, \neg E\} \}$$

une séquence de résolutions correctes conduisant à la clause vide.  $C_8 = Res(C_1, C_7) = \{\neg D, \neg E\}$

$$C_9 = Res(C_6, C_8) = \{\neg E\}$$

$$C_{10} = Res(C_2, C_3) = \{C, E\}$$

$$C_{11} = Res(C_5, C_{10}) = \{E\}$$

$$C_{12} = Res(C_9, C_{11}) = \emptyset$$

justification : on a produit la clause vide à partir de la  $FC(\mathcal{H}_1 \wedge \mathcal{H}_2 \wedge \mathcal{H}_3 \wedge \neg \mathcal{C})$  donc  $FC(\mathcal{H}_1 \wedge \mathcal{H}_2 \wedge \mathcal{H}_3 \wedge \neg \mathcal{C})$  insatisfiable (correction de la méthode de résolution), donc  $\mathcal{H}_1 \wedge \mathcal{H}_2 \wedge \mathcal{H}_3 \wedge \neg \mathcal{C}$  insatisfiable (équivalence sémantique de la forme clausale), donc  $\{\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3\} \models \mathcal{C}$  (théorème d'équivalence des problèmes logiques).

**Question 5 (4 points)** Soit  $H_1, H_2, \dots, H_n, C$  (avec  $n \geq 1$ ) des formules bien formées de la logique des propositions. Montrez que si  $H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$  est insatisfiable alors  $\{H_1, H_2, \dots, H_n\} \models C$ .

$H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$  insatisfiable signifie que pour toute interprétation  $I$  on a  $v(H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C, I) = 0$ . Par la sémantique du  $\wedge$ , on a :

- Soit il existe un  $k \in [1..n]$  tel que  $v(H_k, I) = 0$ ; du coup  $I$  n'est pas un modèle de ce  $H_k$  et donc pas un modèle commun de  $\{H_1, H_2, \dots, H_n\}$  (il ne constraint donc pas  $v(C, I)$  par la définition de la conséquence logique).
- Soit  $v(\neg C, I) = 0$ ; par la sémantique du  $\neg$ , on a donc  $v(C, I) = 1$ ; du coup, même si  $I$  est un modèle commun à  $\{H_1, H_2, \dots, H_n\}$  il est aussi un modèle de  $C$ . CQFD

**Question 6 (4 points)** Soit le syllogisme suivant :

Certains poissons sont urticants

Les raies sont des poissons

Donc certaines raies sont urticantes

1. Modélisez ce raisonnement en logique des prédictats.

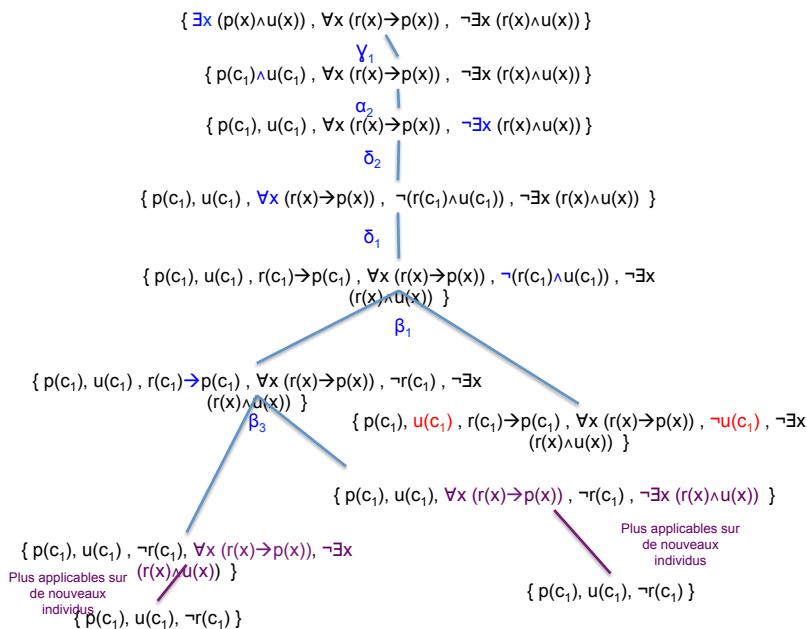
On utilise le vocabulaire logique suivant : Pas de constante et l'ensemble des prédictats contient les prédictats unaires  $p, r, u$  dont l'interprétation intuitive est :

- $p$  : "être un poisson"
- $r$  : "être une raie"
- $u$  : "être urticant"

Le syllogisme se traduit par la conséquence logique suivante :  $\{\exists x (p(x) \wedge u(x)), \forall x (r(x) \rightarrow p(x))\} \models \exists x (r(x) \wedge u(x))$

2. Montrez à l'aide de la méthodes des tableaux que ce n'est pas un raisonnement correct.

un tableau correct. Cf. exemple ci-dessous



3. En déduire une interprétation montrant que ce n'est pas un raisonnement correct.

une interprétation qui rend vraies les hypothèses et fausse la conclusion. Par exemple :  $D = \{c_1\}$ ,  $I(p) = I(u) = \{(c_1, 1)\}$ ,  $I(r) = \{(c_1, 0)\}$ .



Nom :  
Prénom :  
Numéro d'étudiant :

## Examen de seconde session

Tous documents sur support papier autorisés. Durée : 1h30.

L'ensemble des réponses sera à donner sur les feuilles d'énoncé. Ne pas dégrapher les feuilles. Les questions sont indépendantes les unes des autres.

### Question 1.

Proposez un diagramme de classes UML pour décrire la situation suivante. Dans l'énoncé de cette question, les noms des classes que vous devrez organiser sont écrits en italiques. Cela peut nécessiter d'ajouter d'autres classes. Dans une boutique de vente en ligne de livres, on cherche à classer les *Livre* en différentes catégories. Une première classification est réalisée suivant le public visé. Suivant ce critère, on classe les livres en plusieurs sous-catégories disjointes, limitées aux *LivreJeunesse* et aux *LivreAdulte*. Ce public visé se décrit par un intervalle d'âges (minimum et maximum) recommandés pour la lecture. Une seconde classification peut être réalisée suivant le genre littéraire. Elle inclut (la liste n'est pas exhaustive) les *Romans*, les *BandesDessinées* et les *LivresDeVoyages*. Un ouvrage peut appartenir à plusieurs genres. Les *BDjeunesse* sont des bandes dessinées destinées à un jeune public.

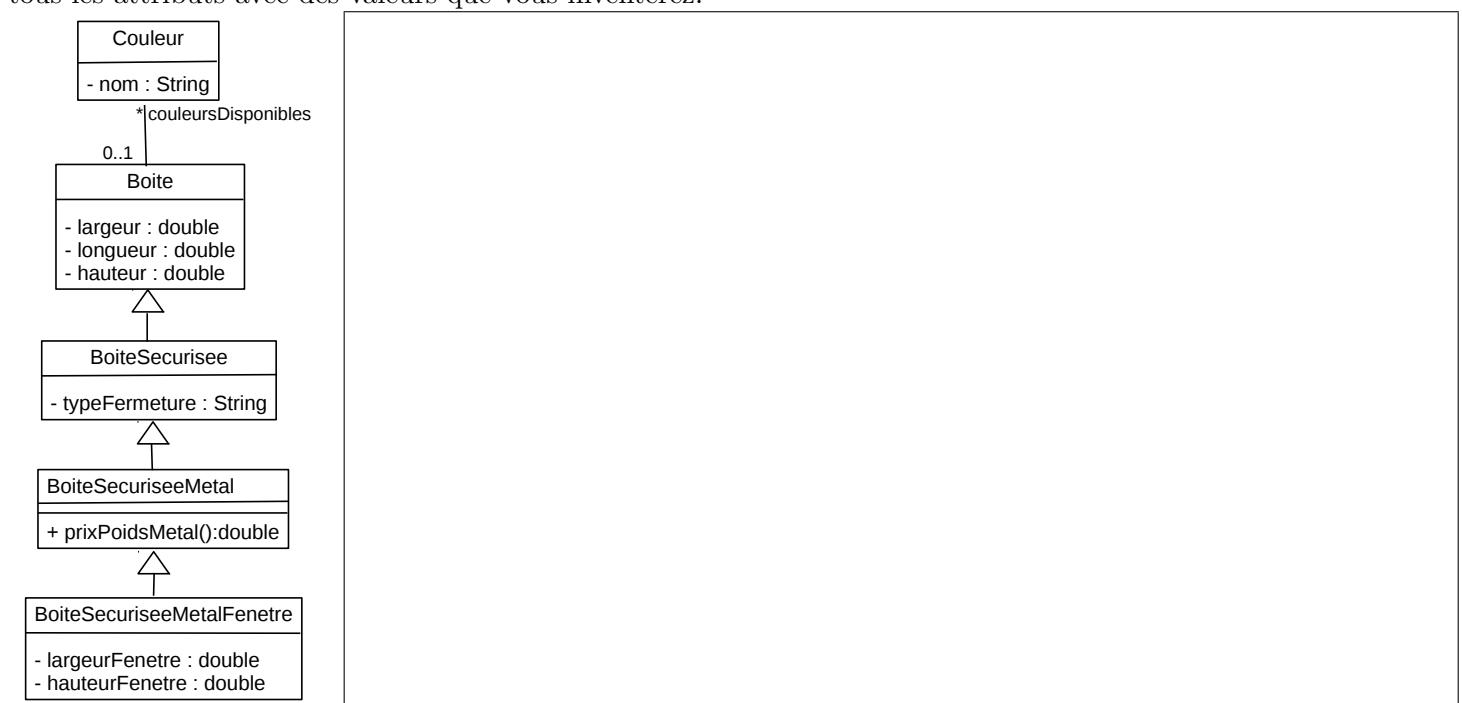
**Question 2.**

Proposez un diagramme de classes UML pour décrire la situation suivante. Dans l'énoncé de cette question, les noms des classes que vous devrez organiser sont écrits en italiques. Cela peut nécessiter d'ajouter d'autres classes. On décrit des articles proposés à la vente dans une pépinière. Un *TypeDePlant* a une référence, un nom d'espèce, un conditionnement (motte terre, godet, pied greffé) et un prix. Un *TypeDeLot* a une référence et contient plusieurs *TypeDePlant*, chacun étant présent dans le lot dans une certaine quantité. Un *TypeDeLot* a un prix qui peut être calculé d'après les *TypeDePlant* qu'il contient et une réduction propre à chaque *TypeDeLot*.

Faites figurer les attributs et les méthodes que vous déduisez de l'énoncé mais pas de constructeurs.

**Question 3.**

La figure ci-dessous vous présente un diagramme de classes décrivant des boîtes. Dessinez une instance de *BoiteSecuriseeMetalFenetre* dont les couleurs disponibles sont le gris et le noir, en donnant une valeur à tous les attributs avec des valeurs que vous inventerez.



**Question 4.**

On vous propose différentes classes dans les listings 1, 2, 3, 4 et 5. Ces classes font partie d'un logiciel destiné à la gestion des tests de langue effectués par les étudiants d'une université.

Listing 1 – Langue.java - représente les différentes langues possibles pour les tests

---

```
package codeSujet;
public enum Langue { franais , anglais , japonais , chinois ;}
```

---

Listing 2 – TableTests.java - représente une table d'association entre un nom de test (par exemple TOEFL-IBT ou JLPT-N3) et la langue associée (respectivement anglais pour le TOEFL ou japonais pour le JLPT)

---

```
package codeSujet;
import java.util.HashMap;

public class TableTests {
    private HashMap<String ,Langue> tableTests = new HashMap<String ,Langue>();
    public void ajouteTest( String nomTest , Langue langue )
        {this.tableTests.put(nomTest , langue);}
    public String connaitreLangueTest( String nomTest )
        {return null;}
}
```

---

Listing 3 – TestLangue.java - représente un test de langue passé par un étudiant (on ne représente pas l'étudiant pour des raisons de simplicité). Un test est dispensé par une école. La classe est associée à une table assurant la correspondance entre noms de tests et langues. Un test a un nom (propre à chaque sous-classe). Un test dispose d'une méthode retournant vrai si et seulement si un niveau suffisant a été atteint par l'étudiant.

---

```
package codeSujet;

public abstract class TestLangue{
    private String ecole;
    private static TableTests tableTests;

    public TestLangue() {}
    public TestLangue( String ecole ) {this.ecole = ecole;}

    public String getEcole() {return ecole;}
    public void setEcole( String ecole ) {this.ecole = ecole;}

    public static TableTests getTableTests() {return tableTests;}
    public static void setTableTests( TableTests tableTests ) {TestLangue.tableTests = tableTests;}

    public abstract String nom();
    public boolean niveauSuffisant(){
        if (this instanceof TOEFLIBT)
        {
            TOEFLIBT t = (TOEFLIBT)this;
            return t.getComprehEcrite() + t.getComprehOrale()
                + t.getExprOrale() + t.getExprEcrite() >= 100;
        }
        if (this instanceof JLPTN3)
        {
            JLPTN3 n = (JLPTN3)this;
            return (n.getLinguistique() + n.getEcrit() + n.getOral()) >= 95
                && n.getLinguistique() > JLPTN3.getNoteEliminatoire()
                && n.getEcrit() > JLPTN3.getNoteEliminatoire()
                && n.getOral() > JLPTN3.getNoteEliminatoire();
        }
        return false;
    }
}
```

---

Listing 4 – JLPTN3.java - représente un test de type JLTP de niveau N3 constitué de trois épreuves.

---

```
package codeSujet;

public class JLPTN3 extends TestLangue {
    private int linguistique , ecrit , oral; // sur 60
```

---

```

private static int noteEliminatoire = 19;
private final static String nom = "JLPT_N3";

public JLPTN3() {}
public JLPTN3(String ecole, int linguistique, int ecrit, int oral) {
    super(ecole);
    this.linguistique = linguistique;
    this.ecrit = ecrit; this.oral = oral;
}

public String nom(){return JLPTN3.nom;}
public int getLinguistique() {return linguistique;}
public void setLinguistique(int linguistique) {this.linguistique = linguistique;}
public int getEcrit() {return ecrit;}
public void setEcrit(int ecrit) {this.ecrit = ecrit;}
public int getOral() {return oral;}
public void setOral(int oral) {this.oral = oral;}
public static int getNoteEliminatoire() {return noteEliminatoire;}
public static void setNoteEliminatoire(int noteEliminatoire) {
    JLPTN3.noteEliminatoire = noteEliminatoire;
}

```

---

Listing 5 – TOEFLIBT.java - représente un test de type TOEFL IBT. Quatre épreuves constituent le test.

```

package codeSujet;

public class TOEFLIBT extends TestLangue {
    private int ComprehEcrite, ComprehOrale,
               ExprOrale, ExprEcrite; // entre 0 et 30
    private final static String nom = "TOEFL_IBT";

    public TOEFLIBT() {}
    public TOEFLIBT(String ecole, int ComprehEcrite, int ComprehOrale,
                    int exprOrale, int exprEcrite) {
        super(ecole);
        this.ComprehEcrite = ComprehEcrite; this.ComprehOrale = ComprehOrale;
        this.ExprOrale = exprOrale; this.ExprEcrite = exprEcrite;
    }

    public String nom(){return TOEFLIBT.nom;}
    public int getComprehEcrite() {return ComprehEcrite;}
    public void setComprehEcrite(int noteComprehEcrite)
        {this.ComprehEcrite = noteComprehEcrite;}
    public int getComprehOrale() {return ComprehOrale;}
    public void setComprehOrale(int noteComprehOrale)
        {this.ComprehOrale = noteComprehOrale;}
    public int getExprOrale() {return ExprOrale;}
    public void setExprOrale(int exprOrale){ExprOrale = exprOrale;}
    public int getExprEcrite() {return ExprEcrite;}
    public void setExprEcrite(int exprEcrite) {ExprEcrite = exprEcrite;}
}

```

---

- a- Ré-écrivez, dans la classe TableTests la méthode connaitreLangueTest(String nomTest) de manière à ce qu'elle retourne la langue associée au test de nom nomTest sous forme chaîne de caractères. Si aucun test n'existe sous ce nom, elle retourne "test inconnu".

b- Etudiez la méthode `boolean niveauSuffisant()` de la classe `TestLangue` (listing 3). Vous pouvez observer que cette méthode réalise un test sur le type de `this` et des type-cast. Ce style n'est pas recommandé en programmation par objets car les responsabilités ne sont pas bien distribuées entre les classes : notamment la classe `TestLangue` manipule, dans cette méthode, des attributs des sous-classes et c'est un code peu extensible. Ré-écrivez en Java cette méthode en adoptant un style plus objet. Soyez précis sur les classes dans lesquelles vous placez les méthodes. Introduisez si possible une méthode auxiliaire spécialisée dans le calcul du nombre de points obtenus.

c- Ecrivez à présent en Java une/des méthode(s) `String resultat()` (dans le style recommandé pour l'objet) :

- Pour tous les tests de langue, la chaîne retournée contient :
  - le nom du test, l'école qui l'a dispensé, la langue concernée, si le niveau suffisant a été atteint et le nombre de points obtenus.
- Pour les TOEFL IBT, la chaîne contient en plus les notes des quatre épreuves (compréhension écrite, compréhension orale, expression écrite, expression orale).
- Pour les JLPT N3, la chaîne contient en plus les notes des trois épreuves (linguistique, écrit, oral).

Indiquez précisément dans quelles classes se trouvent les méthodes que vous écrivez.

d- Le portefeuille de tests de langues d'un étudiant est présenté sous une forme très simple au listing 6.

Listing 6 – PortefeuilleLangues.java.

```
package codeSujet;  
import java.util.Vector;  
  
public class PortefeuilleLangues {  
    private Vector<TestLangue> listeTestsLangues = new Vector<TestLangue>();  
    public PortefeuilleLangues() {}  
}
```

1. Ecrivez une méthode `TestLangue rechercheTest(String nomTest)` qui retourne le test de langue de nom `nomTest` présent dans le portefeuille s'il en existe un, et `null` sinon.

2. Ecrivez une méthode `boolean ajouteTestLangue(TestLangue test)` qui ajoute `test` dans le portefeuille si ce dernier ne contient pas déjà un test portant le même nom. La méthode retourne `true` si l'ajout s'est bien passé, `false` sinon.

3. Mettez en place les éléments de code nécessaires (indiquez clairement quelles classes sont modifiées et comment) pour écrire une méthode de tri des tests de langue du portefeuille d'après leur nom (en suivant l'ordre alphabétique).

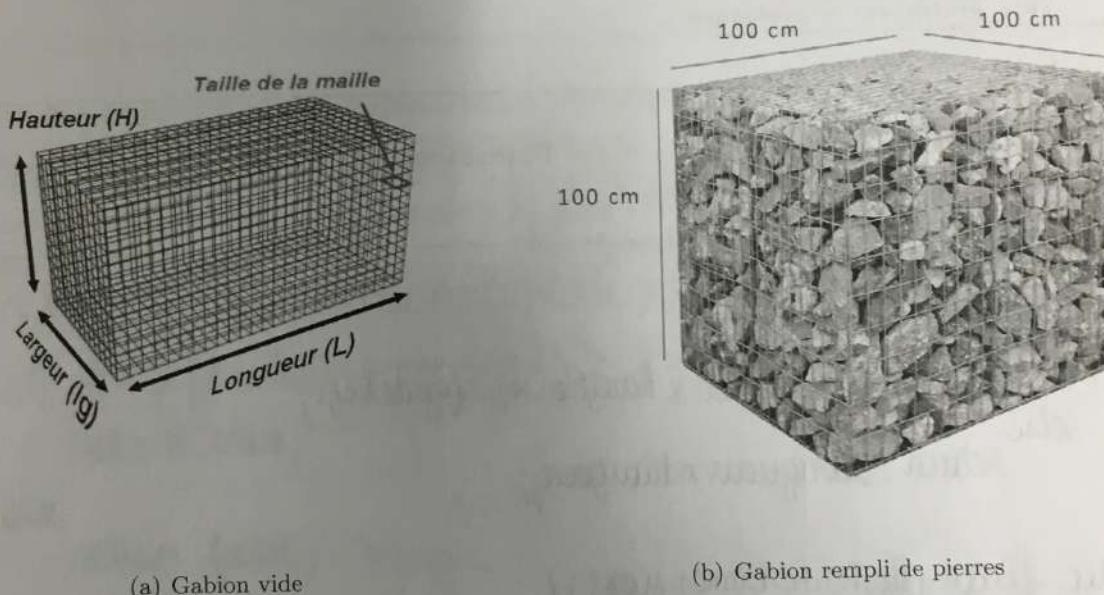
## Contrôle continu

Tous documents sur support papier autorisés. Durée : 1h30.  
L'ensemble des réponses sera à donner sur les feuilles d'énoncé. Ne pas dégrapher les feuilles.

On s'intéresse à un logiciel de simulation de murs de pierres à usage paysager. Toutes les dimensions de longueur sont exprimées en mètres, les volumes en  $m^3$ , les prix en euros.

### 1 Gabions

Dans cette section on s'intéresse aux gabions, qui permettent la construction rapide de murs de pierres. Les gabions sont en effet des sortes de cages métalliques, que l'on peut remplir de pierres (voir figure 1). On se limite dans tout l'énoncé aux gabions et aux pierres de forme parallélépipédique rectangle (ce sont donc des pavés droits).



(a) Gabion vide

(b) Gabion rempli de pierres

FIGURE 1 – Gabion vide (à gauche) et gabion cubique rempli (à droite)

On donne à la figure 2 un diagramme de classes partiel pour les gabions.

La classe **Dimensions** représente les dimensions d'un parallélépipède rectangle (pavé droit) ou de rectangles (dans ce cas la profondeur est nulle). La longueur est toujours la plus grande des trois dimensions (c'est-à-dire que  $\text{longueur} \geq \text{hauteur}$  et  $\text{longueur} \geq \text{profondeur}$ ). La méthode plusPetiteDimension retourne la plus petite des 3 dimensions (c'est-à-dire soit la largeur soit la hauteur).

*profondeur*

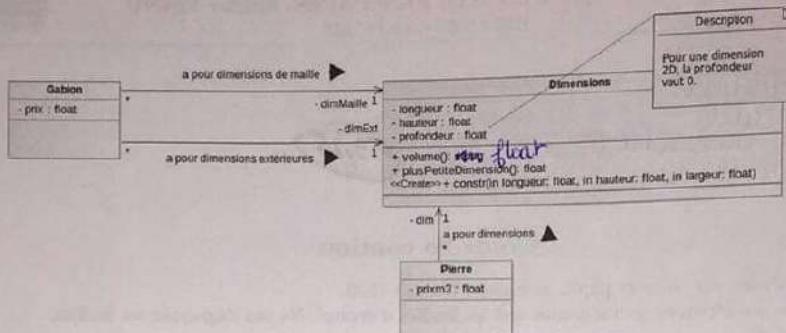


FIGURE 2 – Diagramme de classe partiel pour les gabions et les dimensions

```

public class Dimensions {
    private float longeur;
    private float hauteur;
    private float profondeur;

    public float getLongeur(){
        return longeur;
    }

    public Dimensions(float longeur, float hauteur, float profondeur) {
        this.longeur = longeur;
        this.hauteur = hauteur;
        this.profondeur = profondeur;
    }
}
  
```

Question 1. Donnez en Java pour la classe **Dimensions** la déclaration et l'implémentation des méthodes `volume` et `plusPetiteDimension`.

Réponse à la question 1 :

```

public float volume() { — 0,25
    if (profondeur != 0)
        return longeur * largeur * profondeur;
    else
        return longeur * largeur;
}

public float plusPetiteDimension() { — 0,25
    if (hauteur < profondeur)
        return hauteur;
    else
        return profondeur;
}
  
```

Un gabion a des dimensions extérieures (ce sont les dimensions de la cage métallique) et des dimensions pour les mailles : chaque face d'un gabion est une grille, les mailles sont les "trous" de la grille, elles sont rectangulaires ou carrées. Il est important de mettre dans un gabion des pierres qui sont plus grandes que la taille de la maille, sinon les pierres vont sortir du gabion. On modélise ici des

pierres avec la classe **Pierre**, qui définit juste le prix au  $m^3$  de la pierre, et ses dimensions (on rappelle qu'on se limite ici aux pierres de forme parallélépipédique rectangle (qui sont des pavés droits).

On suppose disposer du code Java suivant :

```
public class Gabion {
    private float prix;
    private Dimensions dimExt;
    private Dimensions dimMaille;

    public Gabion(float prix, Dimensions dimExt, Dimensions dimMaille) {
        this.prix = prix;
        this.dimExt = dimExt;
        this.dimMaille = dimMaille;
    }
    public float getPrix() {
        return prix;
    }
}

public class Pierre {
    private float prixm3;
    private Dimensions dimExt;

    public float getPrixm3() {
        return prixm3;
    }
    public Dimensions getDimExt() {
        return dimExt;
    }
    public Pierre(float prixm3, Dimensions dimExt) {
        this.prixm3 = prixm3;
        this.dimExt = dimExt;
    }
}
```

**Question 2.** Donnez en Java pour la classe **Gabion** l'entête et le corps d'une méthode **estCompatibleAvec** qui, pour une pierre donnée, dit si elle est compatible avec le gabion. On dira qu'une pierre p est compatible avec un gabion g si et seulement si la plus petite dimension de p est une fois et demie plus grande que la longueur de la maille de g.  $\Leftrightarrow$  la plus petite dimension de la pierre est trois fois plus grande que la longueur de la maille

Réponse à la question 2 :

```
public boolean estCompatibleAvec(Pierre p) { 0,5
    if( 3 * p.plusPetiteDimension() > 2 * dimMaille.getLongueur() )
        return true; 1
    else
        return false; 0,25
}
```

(175)

**Question 3.** Donnez en Java le code nécessaire pour déclarer une variable de type **Gabion**, et d'y affecter une nouvelle instance de gabion de dimensions extérieures  $1m \times 1m \times 1m$ , de dimensions de maille  $0.01m \times 0.01m$  et de prix 30 euros.

Réponse à la question 3 :

```
Dimensions dimExt = new Dimensions(1, 1, 1);
Dimensions dimMaille = new Dimensions(0.01, 0.01, 0);
Gabion g = new Gabion(30, dimExt, dimMaille);
```

## 2 Murs

On s'intéresse maintenant aux murs de pierres, une modélisation partielle est donnée à la figure 3. Un mur a des dimensions souhaitées. Un mur peut être ou pas un mur de clôture. On choisit pour un mur le type de pierre à utiliser. Le volume nécessaire de pierres est calculé par la méthode `calculVolumePierre`, on suppose disposer par la suite de l'implémentation de cette méthode. On distingue pour l'instant deux sous-classes de la classe (abstraite) `MurEnPierre : MursAGabion` et `MurMaçonné`. La spécialisation a été faite sur le critère du type de construction. Chaque mur à gabion est lié au type de gabion choisi, et la méthode `nbGabionsNecessaires` calcule et retourne le nombre de gabions nécessaires pour la réalisation du mur, on suppose disposer par la suite de l'implémentation de cette méthode. Chaque mur maçonné a une couleur de mortier.

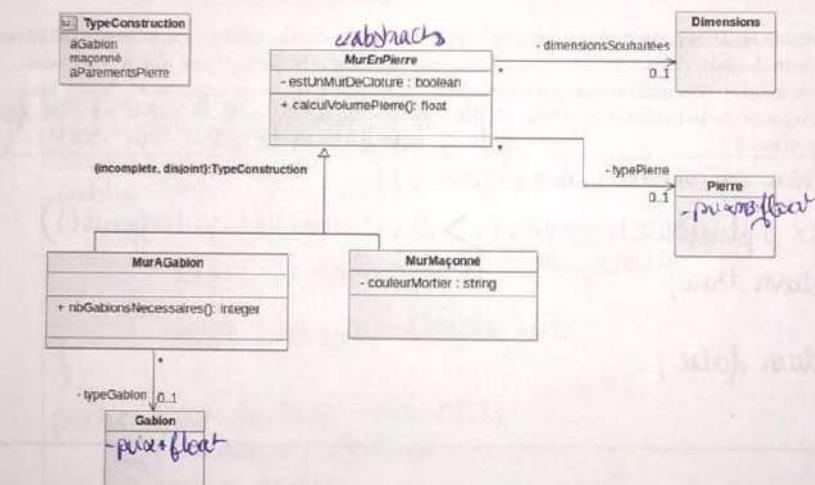


FIGURE 3 – Modélisation partielle des murs

(3)

**Question 4.** On souhaite disposer pour tous les murs d'une méthode `prixMo` qui retourne le prix de la main d'œuvre pour construire le mur. Dans le cas des murs à gabions, ce prix est de x euros par gabion nécessaire (où x est une valeur que l'on fixe ici arbitrairement à 200). Pour les murs maçonnés, le prix est de y euros par  $m^3$  de pierres nécessaires (où y est une valeur que l'on fixe ici arbitrairement à 300). Écrire en Java, partout où elle est nécessaire dans la hiérarchie des murs, cette méthode `prixMo`

(entête et corps). On précisera bien dans quelle classe est placé quel code.  
Réponse à la question 4 :

Dans la classe MurEnPierre :

```
public abstract double prixMo();
```

Dans la classe MurAGabion :

```
public double prixMo() {
    int x = 200;
    return x * nbGabionsNecessaires();
}
```

Dans la classe MurMaconné :

```
public double prixMo() {
    float y = 300;
    return y * super.calculVolumePierre();
}
```

- ③ Question 5. On souhaite disposer pour tous les murs d'une méthode `prixMateriel` qui retourne le prix du matériel nécessaire à la construction du mur. Dans le cas des murs à gabions, il s'agit du prix des pierres nécessaires ajouté au prix des gabions nécessaires. Dans le cas du mur maçoné il s'agit du prix des pierres nécessaires (on néglige le prix du mortier). Écrire en Java, partout où elle est nécessaire dans la hiérarchie des murs, cette méthode `prixMateriel` (entête et corps). On précisera bien dans quelle classe est placé quel code.

Réponse à la question 5 :

Dans la classe MurEnPierre :

```
public abstract double prixMateriel();
```

Dans la classe MurAGabion :

```
public double prixMateriel() {
    return super.calculVolumePierre() * super.typePierre.getPrixM3()
           + nbGabionsNecessaires() * typeGabion.getPrix();
```

}

Dans la classe MurMaconné : 0,75

```
public double prixMateriel() {
```

```
    return super.calculVolumePierre() * super.typePierre.getPrixM3();
```

}

0,75

(2)

**Question 6.** On souhaite disposer pour tous les murs d'une méthode `prix` qui retourne le prix du mur. Pour les murs à gabions comme pour les murs maçonnes, ce prix est la somme du prix de la main d'œuvre et du prix du matériel. Écrire en Java, partout où elle est nécessaire dans la hiérarchie des murs, cette méthode `prix` (entête et corps). On précisera bien dans quelle classe est placé quel code.  
Réponse à la question 6 :

Dans la classe MurEnPierre :

```
public double prix() {  
    return prixMo() + prixMatériel(); }
```

Il n'est pas nécessaire de la recréer dans les sous classes.

(2)

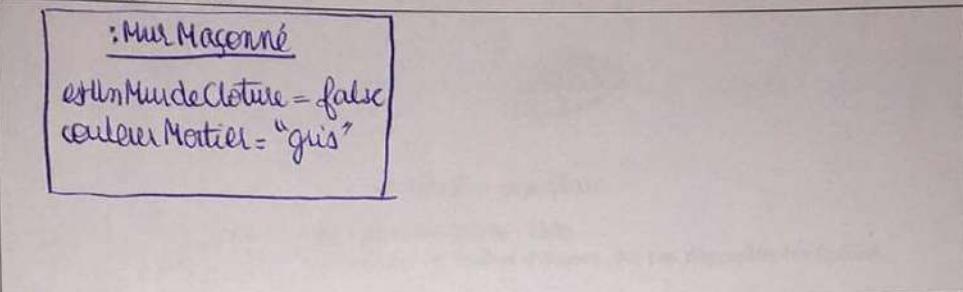
**Question 7.** Ecrire en Java des constructeurs paramétrés pour les classes `MurPierre` et `MurMaçonné`, ces constructeurs n'initialiseront pas la pierre choisie ni les dimensions souhaitées.  
Réponse à la question 7 :

```
public MurEnPierre (boolean b){  
    estUnMurDeClôture = b;  
}
```

```
public MurMaçonné (boolean b, String s){  
    super (b);  
    couleurMatériel = s;  
}
```

**Question 8.** Donnez un diagramme d'instances représentant un mur maconné de mortier gris et qui n'est pas un mur de clôture. Ce mur n'aura pas encore de dimensions souhaitées ni de pierre choisie.  
Réponse à la question 8 :

A1



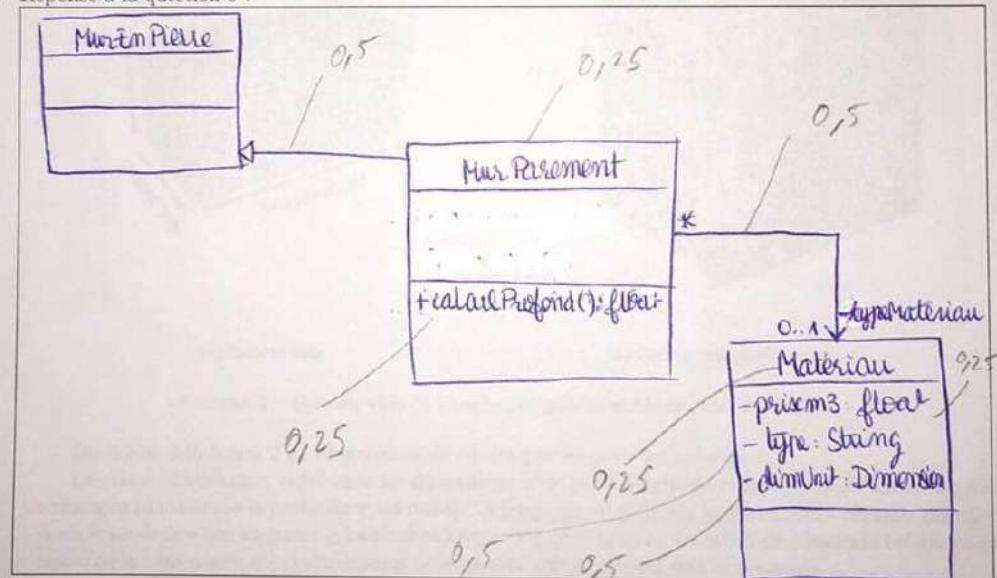
### 3 Murs à parements de pierre

On souhaite ajouter à la modélisation existante celle des murs à parement de pierre. Ces murs sont des murs de pierre qui sont construits dans un matériau autre que la pierre, et qui sont juste recouverts de pierres ensuite (les pierres sont en quelque sorte collées sur le matériau avec du mortier). Ces murs sont décrits par le matériau central utilisé. De tels murs peuvent être à mortier apparent ou pas. Le matériau est décrit par son prix par  $m^3$  et son type (brique ou parpaing), ainsi que par sa dimension unitaire (dimension d'une brique ou dimension d'un parpaing). Pour ces murs, on peut calculer sa profondeur approximative en ajoutant deux fois la profondeur de la pierre à la profondeur du matériau (on néglige la profondeur du mortier).

**Question 9.** Donnez en UML une modélisation des murs à parement de pierre.

(3)

Réponse à la question 9 :



Semestre 4

## CORRECTION CC INFO

Q1/ 1,5

```
public float volume() {
    return longueur * hauteur * profondeur; // ou getLongueur() * getHauteur()
                                                * getProfondeur(),
}
```

```
public float plusPetiteDimension() {
    if (hauteur >= profondeur)
        return profondeur;
    else
        return hauteur;
}
```

Q2/ 1,75

```
public boolean estCompatibleAvec (Pierre p) {
    return p.getDimEAT().plusPetiteDimension() /
        this.dimMaille.getLongueur() >= 1,5 ;
}
```

Q3/ 1,75 Gabien g = new Gabien (30f, new Dimensions (1,1,1),
new Dimensions (0.01f, 0.01f, 0));

```
// ou Dimensions eat = new Dimensions (1,1,1);
// Dimensions maille = new Dimensions (0.01f, 0.01f, 0);
// Gabien g = new Gabien (30f, eat, maille);
```

Q4/ 1,3

```
// Dans MurEnPierre
public abstract float prixMo();
```

```
// Dans MurGabien
public float prixMo() {
    return 200 * nbGabionsNecessaires();
}
```

// Dans MurMaconné  
public float prixMo() {  
 return 300 \* calculVolumePierre();  
}  
} <sup>super</sup>  
juste mais inutile

4

Q5 // Dans MurEnPierre  
public float prixMateriel() {  
 return calculVolumePierre() \* typePierre.getPrixM3();  
}

// Dans MurAGabion  
public float prixMateriel() {  
 return super.prixMateriel() + mGabionsNecessaires \* typeGabion.getPrix();  
}

// Dans MurMaconné  
// rien → on appelle celle de la superclasse qui est suffisante ici

1/2

Q6 // Dans MurEnPierre  
public float prix() {  
 return prixMo() + prixMateriel();  
}

// rien ailleurs

1/2/5

Q7 public MurEnPierre(boolean estCloture) {  
 estUnMurDeCLOTURE = estCloture;  
}

```

public MurMaconné ( boolean estCloture, String couleur )
    super ( estCloture );
    couleurMortier = couleur;
}

```

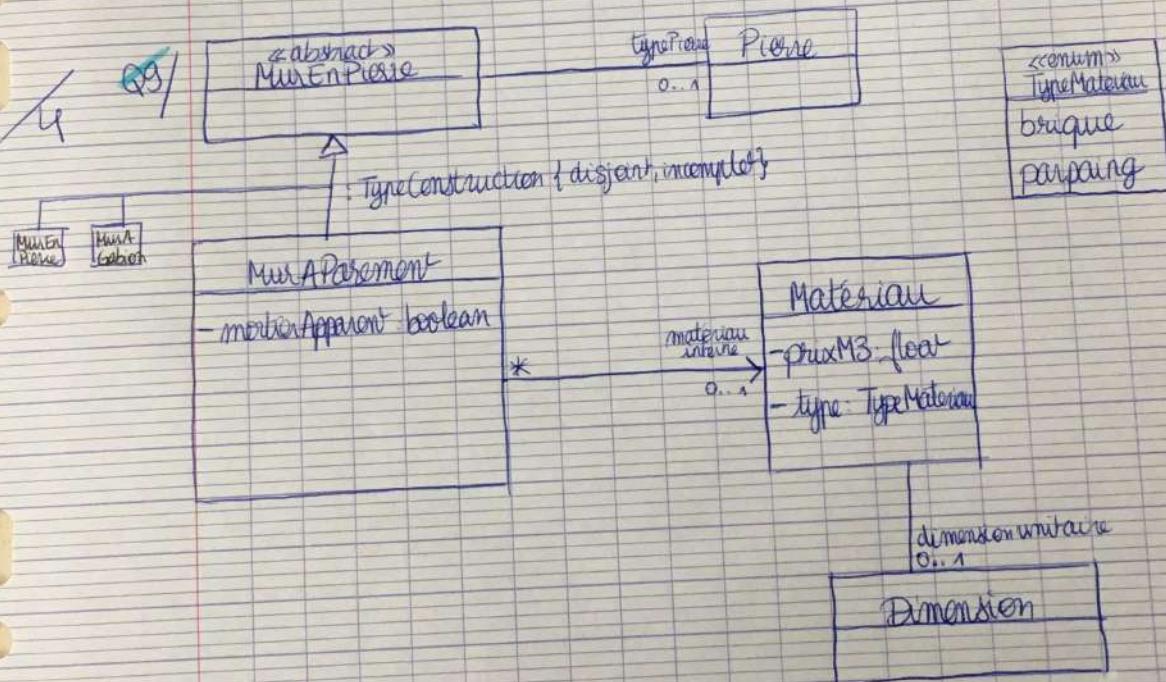
15

Q8

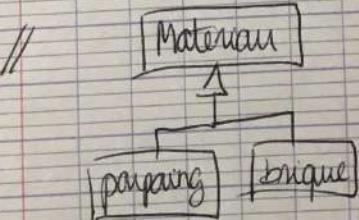
Mur Mâconné  
couleurMortier = "gris"  
estUnMurDeCloture = false

4

Q9



// on aurait pu mettre





Nom :  
Prénom :  
Numéro d'étudiant :  
Groupe :

### Contrôle continu 3

Tous documents sur support papier autorisés. Durée : 1h15  
L'ensemble des réponses sera à donner sur les feuilles d'énoncé. Ne pas dégrapher les feuilles.

La figure 1 vous présente un extrait d'une représentation des fichiers dans un système d'exploitation générique. L'interface `Int_Fichier` expose les opérations disponibles sur tous les fichiers. La classe `Abs_Fichier` comporte les informations et les comportements (parties de codes de méthodes) communs à tous les fichiers. Les objets de `Abs_Fichier` sont divisés selon le critère `TypeFichier` qui indique si un fichier est un conteneur (par exemple un répertoire) ou non. La classe `FichierSimple` représente les fichiers ordinaires (textes, images, liens symboliques, etc.). Ceux-ci sont décrits par une taille (en octets) et une extension (telle que "jpg", "txt", "odg", etc.). La classe `Répertoire` représente les répertoires. Un répertoire contient des fichiers.

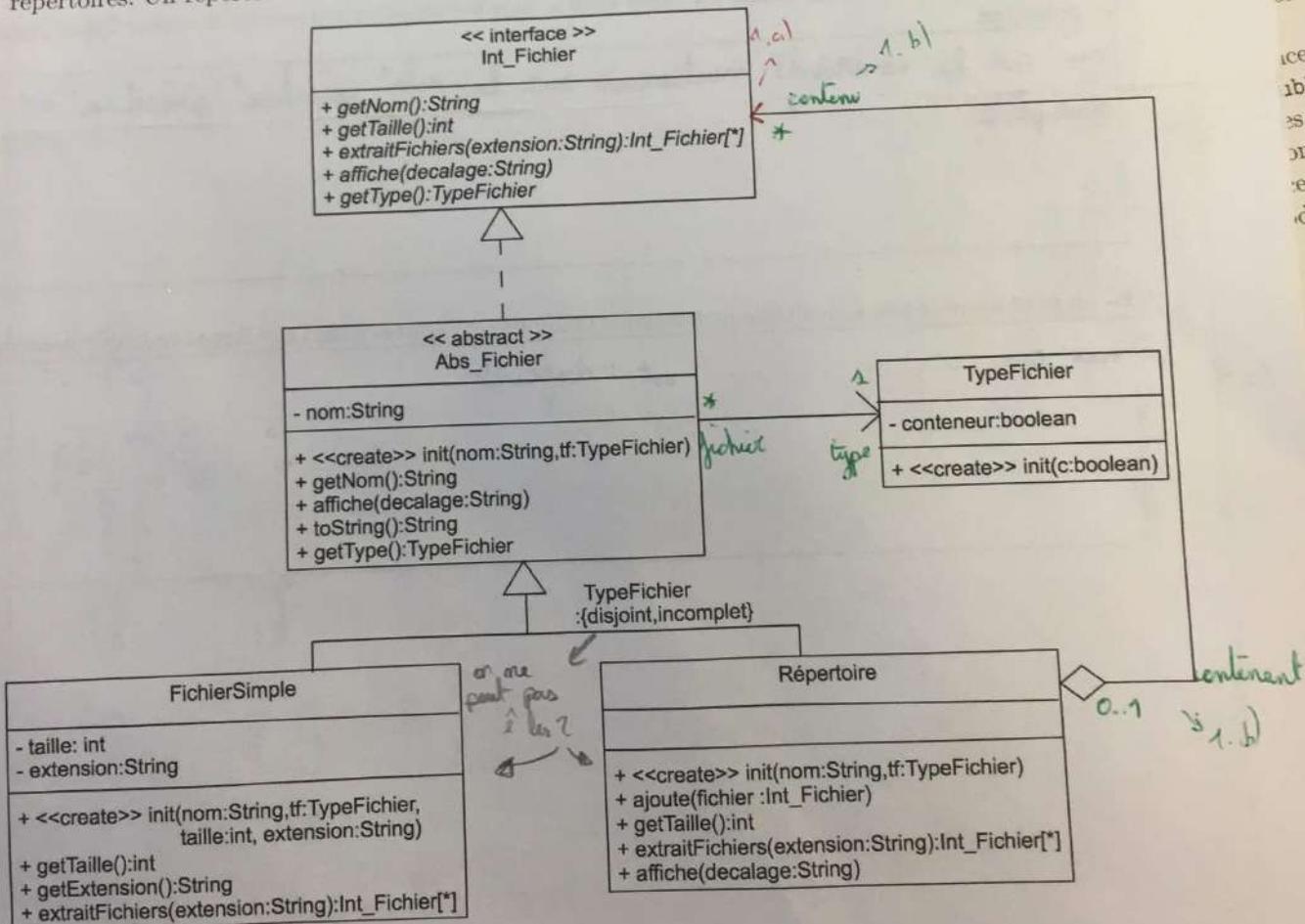
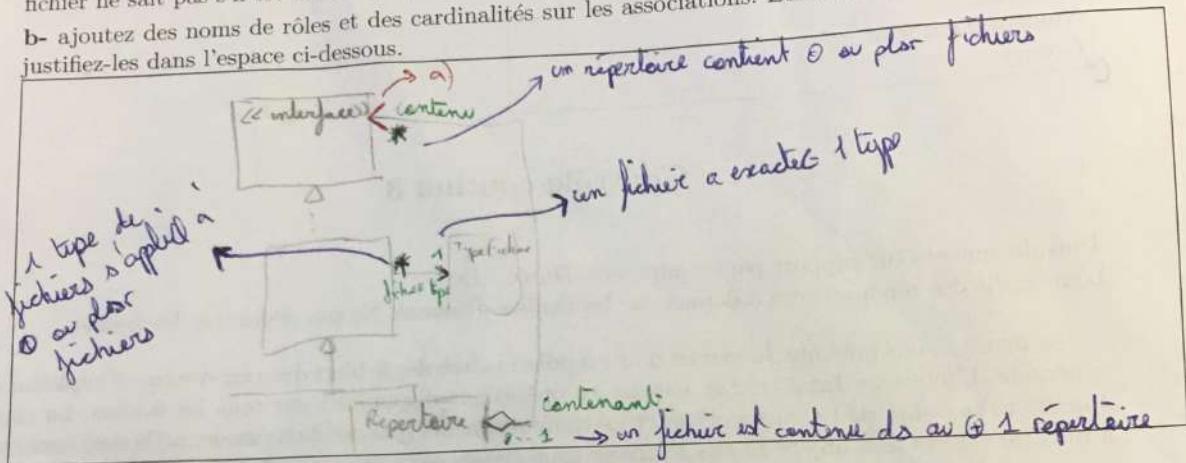


FIGURE 1 – Diagramme de classes à compléter

**Question 1.** Complétez le diagramme par les informations suivantes :

- a- Représentez sur le diagramme le fait qu'un répertoire connaît les fichiers qu'il contient, mais qu'un fichier ne sait pas s'il est dans un répertoire.  
 b- ajoutez des noms de rôles et des cardinalités sur les associations. Ecrivez-les sur le diagramme et justifiez-les dans l'espace ci-dessous.



**Question 2.** En vous appuyant sur les contraintes indiquées sur le groupe de relations de spécialisation/généralisation, précisez si :

- a- on peut ajouter une sous-classe « Fichier Lien » dans la spécialisation suivant le critère TypeFichier et pourquoi.

oui car la contrainte mentionnée sur la rela° spécifia°/généralisa° est :  
 incomplète

- b- on peut ajouter une sous-classe instanciable commune à FichierSimple et Répertoire et pourquoi.

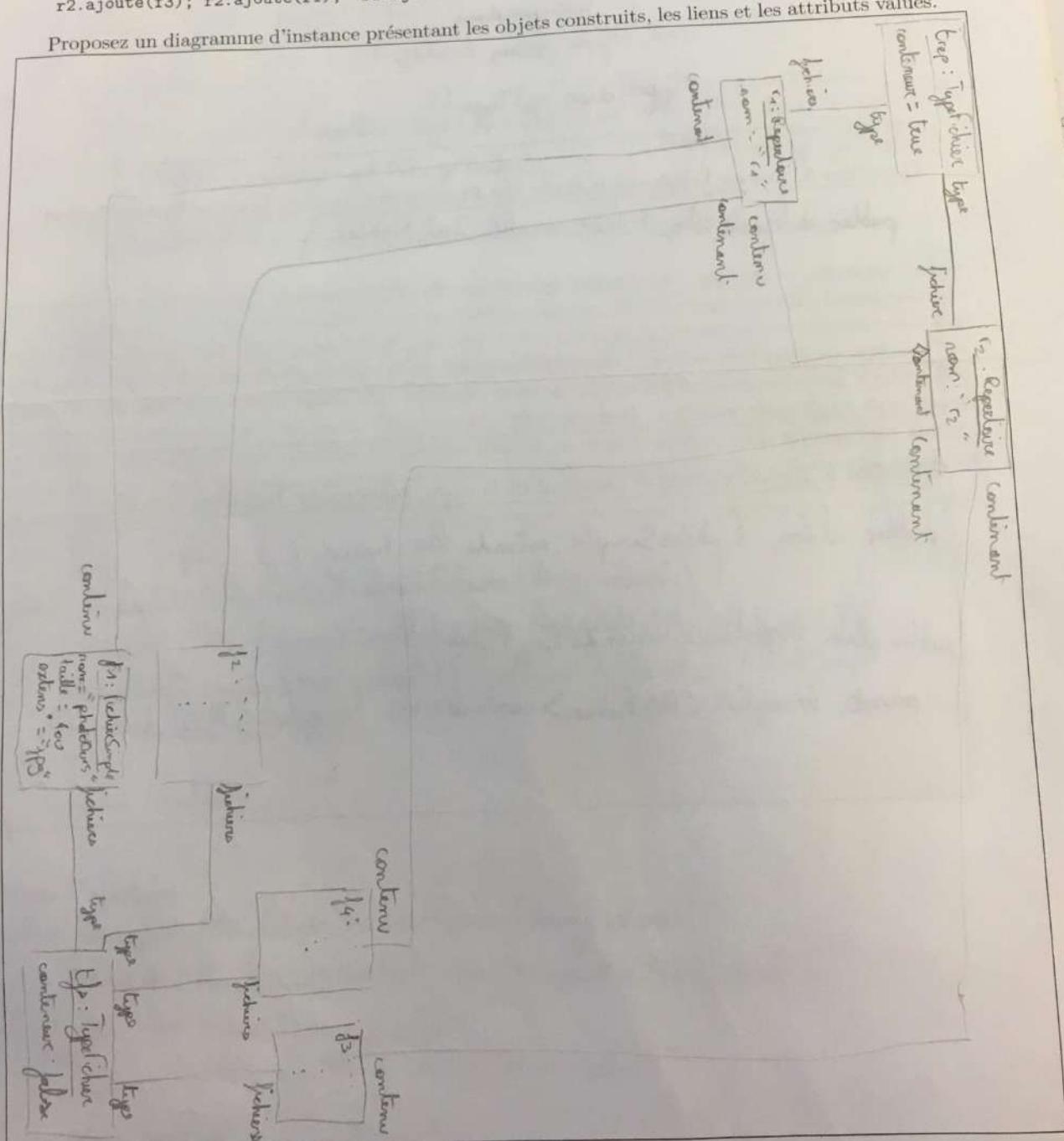
non car ... est : disjoint

**Question 3.** Etant donné les instructions suivantes (les constructeurs sont construits de manière classique et suivent l'ordre de déclaration sur le diagramme) :

```

13
TypeFichier tfs = new TypeFichier(false);
TypeFichier trep = new TypeFichier(true);
Int_Fichier f1 = new FichierSimple("photoOurs",tfs,400,"jpg");
Int_Fichier f2 = new FichierSimple("photoFleur",tfs,300,"png");
Int_Fichier f3 = new FichierSimple("photoOcean",tfs,500,"png");
Int_Fichier f4 = new FichierSimple("photoEcume",tfs,200,"JPG");
Repertoire r1 = new Repertoire("r1",trep);
r1.ajoute(f1); r1.ajoute(f2);
Repertoire r2 = new Repertoire("r2",trep);
r2.ajoute(f3); r2.ajoute(r1); r2.ajoute(f4);
  
```

Proposez un diagramme d'instance présentant les objets construits, les liens et les attributs valued.



en Java

Question 4. Ecrivez l'interface Int\_Fichier.

```

public interface Int_Fichier {
    public String getName();
    abstract
    public abstract int getSize();
    public abstract ArrayList<Int_Fichier> subFiles(String extension);
    —————— void affich(String decalage);
    —————— TypeFichier getType();
}

```

Question 5. Ecrivez l'entête de la classe Abs\_Fichier et l'entête de la classe FichierSimple.

```

public abstract Abs_Fichier implements Int_Fichier {
}

```

Question 6. Ecrivez l'entête et l'attribut de la classe Repertoire. Initialisez l'attribut.

```

public class FichierSimple extends Abs_Fichier {
}

```

```

public class Repertoire extends Abs_Fichier {
    private ArrayList<Abs_Fichier> contenu = new ArrayList<Abs_Fichier>();
}

```

**Question 7.** Ecrivez la méthode `getTaille` dans les classes où elle vous semble utile (appuyez-vous sur le diagramme). Pour les fichiers simples, elle retourne la valeur de l'attribut. Pour les fichiers répertoires, elle correspond à la somme des tailles des éléments contenus, à tous les niveaux d'imbrication. Par exemple, pour les objets créés précédemment, la taille de `r2` est 1400. Indiquez clairement dans quelle classe vous mettez quel code.

Dans `FichierSimple` : public int `getTaille()` {  
    return taille;  
}

Dans `répertoire` : public int `getTaille()` {  
    int résultat = 0;  
    for (int i=0; i < contenu.size(); i++) { } or for (int fichier : contenu) {  
        résultat += contenu.get(i).getTaille();  
    }  
    return résultat;  
}

**Question 8.** Ecrivez la méthode `extraitFichier(String extension)`, partout où elle vous semble utile (appuyez-vous sur le diagramme). Pour les fichiers simples, elle retourne une liste contenant l'objet courant s'il a l'extension requise (sinon une liste vide). Pour les répertoires, elle retourne une liste contenant tous les fichiers du répertoire et de ses sous-répertoires jusqu'au plus profond niveau d'imbrication, qui ont l'extension requise. Par exemple, pour les objets créés précédemment, l'instruction `r2.extraitFichier("jpg")` retourne `[photoOurs, photoEcume]`. Indiquez clairement dans quelle classe vous mettez quel code. Suivant la manière dont vous vous y prenez, vous pouvez être amenés à fusionner des listes, pour cela vous pouvez utiliser la méthode `addAll(List<..> autreListe)` des classes `ArrayList` ou `Vector` (qui sont des `List`).

- Dans `Fichier Simple` :  
public ArrayList<Abs\_Fichier> `extraitFichiers(String extens°)` {  
    ArrayList<Abs\_Fichier> résultat = new ArrayList<Abs\_Fichier>();  
    if (extens°.equals(this.extension)) { attribut de la classe Fichier Simple  
        résultat.add(this);  
    }  
    return résultat;  
}

- Dans `Répertoire` :  
public ArrayList<Abs\_Fichier> `extraitFichiers(String extens°)` {  
    ArrayList<Abs\_Fichier> résultat = new ArrayList<Abs\_Fichier>();  
    for (Int\_Fichier i : contenu) {  
        résultat.addAll(i.extraitFichiers(extens°));  
    }  
    return résultat  
} → une `ArrayList<Int_Fichier>`

**Question 9.** Ecrivez la méthode `affiche()` qui affiche sur la console un fichier, partout où elle vous semble utile (appuyez-vous sur le diagramme). S'il s'agit d'un répertoire, le contenu est affiché de manière indentée. Par exemple, pour les objets créés précédemment l'affichage sera le suivant :

```
r2
photoOcean
r1
photoOurs
photoFleur
photoEcume
```

Do Fichier : public void affiche (String decalage) {
 System.out.println (decalage + nom);
}

Do Répertoire :
 public void affiche (String decalage) {
 super.affiche (decalage);
 for (int i = 0; i < contenue.length; i++) {
 if (contenue[i] instanceof Fichier) {
 ((Fichier) contenue[i]).affiche (decalage + " ");
 } else {
 ((Répertoire) contenue[i]).affiche (decalage + " ");
 }
 }
 }
}



Examen terminal écrit : Session 1

Date : 15 mai 2019

Mention Informatique

Licence 2<sup>ème</sup> année : Programmation applicative (HLIN403)

Durée de l'épreuve : 2 heures

Documents autorisés : tous

Les documents et les calculatrices sont autorisés (ce qui veut dire que la réponse à une question de cours doit montrer que vous avez *compris* la notion). Lisez l'ensemble du sujet. N'oubliez pas d'inscrire les informations demandées sur vos copies et relisez-vous 5 minutes avant de rendre votre copie. N'inscrivez aucun signe distinctif sur votre copie. Bonne chance !

## 1 Structure de données abstraites (10 points)

**Exercice 1.** (2 points) Rappeler brièvement ce qu'est une structure de données abstraite et en donner un exemple, en précisant les différents types d'éléments de l'interface.

On se propose d'implémenter un jeu de plateau se déroulant sur une grille en 2D. Deux équipes s'affrontent : les Bons et les Méchants. Chaque équipe possède un certain nombre de pions. Chaque pion se caractérise, à chaque tour de jeu, par :

- des coordonnées (entières) sur la grille (donc une abscisse et une ordonnée) ;
- un nombre de points de vie (100 au départ)
- un nombre de points de dégats (10 au départ, et 5 si le pion a moins de 25 points de vie)

**Exercice 2.** (3 points) Proposer une implémentation pour la structure de données de pion. Vous préciserez bien votre choix pour cette implémentation, ainsi que l'interface permettant de la manipuler.

On implémente une équipe à l'aide d'une liste de pions. Voici l'interface pour cette donnée :

Constructeurs	Prédicats
-----	-----
equipe-vide	equipe-vide?
make-equipe (list)	
ajouter-pion (cons)	
Accessseurs	Couche métier
-----	-----
premier-pion	enlever-pion
pions-suivants	

**Exercice 3.** (2 points) Quels sont les prototypes (paramètres d'entrée, résultat en sortie) des différents éléments de l'interface d'équipe ?

**Exercice 4.** (2 points) Implémentez la fonction `enlever-pion`, en n'utilisant que les éléments de l'interface. Quel est l'intérêt de cette démarche ?

**Exercice 5.** (1 point) Proposez une autre implémentation possible pour cette structure de données (on ne demande pas de ré-implémenter l'interface précédente).

## 2 Récursivité (7 points)

Un tour de jeu se passe en deux phases :

- une phase de combat
- une phase de déplacement

La phase de combat consiste à se faire combattre des pions qui sont dans un même périmètre (c'est-à-dire qui sont dans un même carré de deux cases de côté maximum), voir figure 1. Le combat entre deux pions consiste à enlever les points d'attaque de l'un de pions aux point de vie de l'autre pion, et inversement. Si le pion est trop affaibli, sa capacité d'attaque diminue. Si le nombre de points de vie d'un pion devient négatif ou nul, il est retiré de l'équipe.

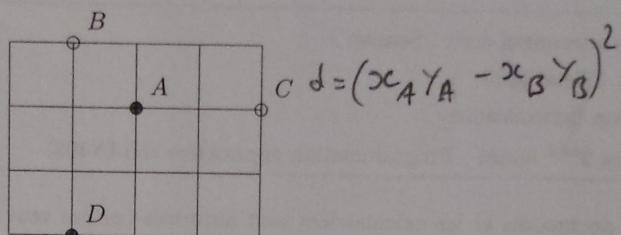


FIGURE 1 – Les Bons sont en gris foncé, les Méchants en gris clair. Le pion A combat le pion B et le pion C, mais les pions B et D sont trop loin pour se combattre.

**Exercice 6.** (1 point) Implémenter la fonction `combat_singulier` qui prend en paramètre deux pions, et renvoie une liste de pions correspondant aux pions en paramètre une fois qu'ils ont combattu l'un contre l'autre (les pions ne sont pas modifiés s'ils sont trop loin pour combattre).

**Exercice 7.** (1 point) Implémenter la fonction `combat_equipe` qui prend en paramètre une équipe et un pion, et renvoie une paire constituée de l'équipe et du pion après la phase de combat du pion avec chaque élément de l'équipe.

**Exercice 8.** (1 points) Implémenter la fonction `combat_total` qui prend en paramètre deux équipes, et renvoie une paire contenant les deux équipes après un tour de combat.

La phase de déplacement consiste, pour chaque pion, à choisir une direction aléatoirement parmi les 4 possibles dans la grille, et à modifier ses coordonnées.

**Exercice 9.** (2 points) Écrire la fonction `déplacement` qui prend en paramètre une équipe et renvoie une équipe où chacun des pions a subi un déplacement aléatoire. Deux versions sont demandées :

- en récursivité enveloppée
- en récursivité terminale (explicitez quelle version est laquelle).

La fonction `random` prend un paramètre entier  $n$  et renvoie un nombre aléatoire entre 0 et  $n - 1$ .

On rappelle la fonction `map` qui prend en paramètre une fonction et une liste, et renvoie une liste correspondant à l'application de la fonction à chaque élément de la liste en entrée. Exemple :

```
> (map (lambda (x) (+ 1 x)) '(1 2 3))
(2 3 4)
```

On ne demande PAS d'implémenter cette fonction.

**Exercice 10.** (2 points) Quel est l'intérêt d'utiliser ces fonctions génériques ? Réécrire la fonction `déplacement` en utilisant cette fonction.

### 3 Dataflow (3 points)

**Exercice 11.** (1 point) Expliquez les grandes lignes de la programmation dataflow. Quelles sont les deux formes spéciales qui permettent de l'implémenter en scheme ? Expliquez leurs actions.

**Exercice 12.** (2 points)

1. Construire le flux de la suite des cubes entiers.
2. Que fait le flux suivant :

```
(define mysterious-flow
  (stream-filter (lambda (x) (= 3 (modulo x 7)))
    integers))
```

où `integers` est le flux des entiers dans l'ordre croissant à partir de 1.



Examen terminal écrit : Session 1

Date : 12 mai 2016

Mention Informatique

Licence 2<sup>ème</sup> année : Programmation applicative (HLIN403)

Durée de l'épreuve : 2 heures

Documents autorisés : tous

Les documents et les calculatrices sont autorisés (ce qui veut dire que la réponse à une question de cours doit montrer que vous avez *compris* la notion). Lisez l'ensemble du sujet. Les différentes sous-parties peuvent être traitées indépendamment. N'oubliez pas d'inscrire les informations demandées sur vos copies et relisez-vous 5 minutes avant de rendre votre copie. N'inscrivez aucun signe distinctif sur votre copie. Bonne chance !

## 1 Récursivité terminale

**Exercice 1.** (2 points) Rappelez brièvement ce qu'est la récursivité terminale par rapport à la récursivité enveloppée. Donnez un exemple pour chacune.

**Exercice 2.** (2 points) Écrire en récursivité enveloppée une fonction count qui prend en paramètres un élément et une liste et compte le nombre d'occurrences de l'élément dans la liste. Exemples :

```
> (count 'a '(a b (a a c) (c (d a) e) d))  
4  
> (count 'a '(b c d (e f)))  
0
```

**Exercice 3.** (2 points) Réécrire la fonction précédente en récursivité terminale, en expliquant les changements que vous avez appliqués par rapport à l'écriture de l'exercice précédent.

## 2 Récursivité arborescente

On dispose de la structure de donnée d'arbre suivante : un arbre est une paire composée d'un élément et d'une liste d'arbres représentant les fils du nœud actuel. L'interface disponible est la suivante (on ne demande pas de l'implémenter) :

Constructeur

`make-arbre e l`      Construit l'arbre de racine `e` et ayant pour fils les éléments de la liste `l`.

Accesseurs

`element a`      Renvoie la valeur de la racine de l'arbre `a`  
`fils a`      Renvoie la liste des fils de la racine de l'arbre `a`.

Prédicats

`arbre-vide? a`      Renvoie vrai si `a` est un arbre vide.  
`feuille? a`      Renvoie vrai si `a` est réduit à une feuille.

**Exercice 4.** (3 points) Écrire une fonction effeuillage qui prend en paramètre un arbre et lui enlève toutes ses feuilles. Par exemple :

```
> (effeuillage '(1. ((2. ((3. ()) (4. ((5. ()) (6. ()))))))))  
(1. ((2. ((4. ())))))   
⇒ '(1. (2. (4. ()))) 1. (2. ()))
```

## 3 Dataflow

**Exercice 5.** (1 point) Expliquez les grandes lignes de la programmation dataflow. Quelles sont les deux formes spéciales qui permettent de l'implémenter en schéma ? Expliquez leurs actions.

**Exercice 6.** (2 points)

1. Construire le flux de la suite des carrés entiers.
2. Que fait le flux suivant :

```
(define mysterious-flow  
  (stream-filter (lambda (x) (not (= 0 (modulo x 7))))  
    integers))
```

où `integers` est le flux des entiers dans l'ordre croissant à partir de 1.

## 4 Abstraction sur les fonctions

**Exercice 7.** (2 points) Étant donnée une fonction  $f$ , on veut construire une fonction qui calcule une approximation de la dérivée de  $f$ . Pour  $dx$  "petit", on définit la fonction dérivée  $f'$  par :

$$f' : x \mapsto \frac{f(x + dx) - f(x)}{dx}$$

Écrire la fonction `derive` qui prend une fonction  $f$  en paramètre et une valeur pour  $dx$ , et renvoie la fonction  $f'$ . Exemple :

```
>(define cube (lambda (x) (* x x x)))
>(cube 10)
1000
>((derive cube 0.0001) 10)
300.0030000893
```

## 5 Structures de données abstraites

Une rotation autour de l'origine du plan en deux dimensions est représentable par une matrice à deux lignes et deux colonnes dont le déterminant est égal à 1 (pour simplifier). Voici quelques exemples de matrice de rotation :

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \quad \begin{pmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix} \quad \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

FIGURE 1 – Quelques exemple de rotations dans le plan, de centre l'origine et d'angle (en radian) respectivement, de gauche à droite : 0 (rotation identité),  $\pi$ ,  $\frac{\pi}{3}$  et un angle quelconque  $\theta$ .

Plus généralement, on représente ces matrices sous la forme :

$$\begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix}$$

Voici l'interface souhaitée pour la structure de données `Rotation` :

### Constructeur

`make-rotation a b c d` Construit la rotation correspondant à la matrice  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$

### Accesseurs

<code>get-x11 rot</code>	Renvoie la valeur de la case en haut à gauche de la matrice de rotation <code>rot</code>
<code>get-x12 rot</code>	Renvoie la valeur de la case en haut à droite de la matrice de rotation <code>rot</code>
<code>get-x21 rot</code>	Renvoie la valeur de la case en bas à gauche de la matrice de rotation <code>rot</code>
<code>get-x22 rot</code>	Renvoie la valeur de la case en bas à droite de la matrice de rotation <code>rot</code>

### Prédicats

`rotation? rot` Renvoie vrai si `rot` est une matrice de rotation

**Exercice 8.** (2 points) Voici l'implémentation qui a été choisie pour le constructeur :

```
(define (make-rotation a b c d)
  (cons (cons a b) (cons c d)))
```

De quel type d'objet s'agit-il ? Donnez la représentation sous forme de boîtes et pointeurs de l'objet défini par :

```
(define m (make-rotation 1 0 0 1))
```

**Exercice 9.** (3 points) Implémenter l'interface pour la structure de données `Rotation`. Pour le prédictat `rotation`, on rappelle (pour simplifier) qu'une matrice est une rotation si elle satisfait les deux conditions suivantes :

- elle a la bonne structure (en termes de paires)

- son déterminant vaut 1. Rappel : le déterminant d'une matrice  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$  est donné par  $ad - bc$ .

**Exercice 10.** (1 point) Modifiez le constructeur donné ci-dessus pour qu'il vérifie si le déterminant vaut 1 avant de créer la matrice (et renvoie un message d'erreur si ce n'est pas le cas).



UNIVERSITE DE MONTPELLIER  
FACULTE DES SCIENCES



Session : 1	Durée de l'épreuve : ..... 2 ..... heures
Date : ... 19/05	Documents autorisés : ..
..... <input type="checkbox"/>	Une page recto verso manuscrite
Licence X	.....
Mention : ...Informatique.....	Matériels autorisés : ...Aucun....
.....	.....
Parcours : .....	.....

### HLIN401 Algorithmique et complexité

#### Exercices.

1. 1 point

Dessiner tous les tas possibles (minimum à la racine) avec les cinq nombres {1, 2, 3, 4, 5}

2. 4 points

Ecrire le constructeur par recopie d'un arbre binaire.

Vous prendrez soin de remplir correctement le champs *Pere* de chacun des noeuds de l'arbre recopié.

3. 3 points

Un programme s'exécute sur une donnée de taille  $n_1$  en un temps  $t_1$ .

Le même programme s'exécute sur une donnée de taille  $n_2 = 2n_1$  en un temps  $t_2$ .

Quel sera le temps  $t_3$  de l'exécution de ce programme sur une donnée de taille  $n_3 = 3n_1$  dans chacun des cas suivants :

**cas i**  $t_2 = 4t_1$  (et ce quel que soit  $n_1$ )

**cas ii**  $t_2 = t_1^2$  (et ce quel que soit  $n_1$ )

**cas iii**  $t_2 = t_1 + 2\log_2(2)$ . (et ce quel que soit  $n_1$ ) ; on rappelle que  $\log_2(2) = 1$ .

Justifiez en terme d'analyse de complexité.

## Problème

### Préalable

Pour manipuler un ABR  $A$ , on utilisera les primitives suivantes (de complexité dans  $\theta(1)$ ) :

- $VideP(A)$  qui renvoie un booléen (vrai si et seulement si  $A$  est vide)
- $FeuilleP(A)$  qui renvoie un booléen (vrai si et seulement si  $A$  est réduit à une feuille).
- $SAG(A)$  et  $SAD(A)$  qui renvoient les sous ABR (éventuellement vides) respectivement gauche et droite de l'ABR (non vide)  $A$ .
- $Val(A)$  qui renvoie la valeur stockée à la racine de l'ABR (non vide)  $A$ .

Toutes les complexités demandées devront être fournies si possible en fonction de la hauteur  $h$  de l'ABR et sinon en fonction de son nombre  $n$  de sommets.

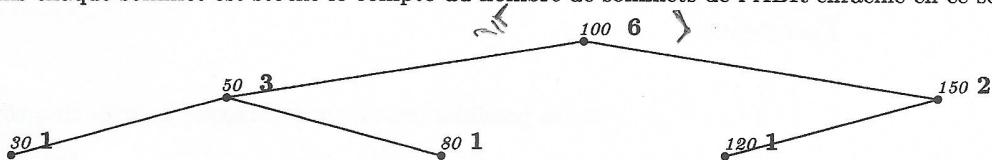
#### Question 1 4 points

Écrire l'algorithme récursif  $NbreSommet(A)$  qui renvoie le nombre de sommets d'un ABR  $A$ .

Donner la classe de complexité de cette algorithme avec un minimum de justifications.

### Un ABR amélioré

Dans chaque sommet est stocké le compte du nombre de sommets de l'ABR enraciné en ce sommet.



En gras apparaît le compte du nombre de sommets de l'ABR enraciné dans le sommet correspondant. Les petites valeurs en italique sont celles des sommets de l'ABR.

On dispose des primitives (de complexité dans  $\theta(1)$ )

- `void Incrementer(A)` augmente de 1 le compte du nombre de sommets stocké dans la racine de  $A$   
Cette primitive n'est définie que pour un ABR non vide  $A$ .
- `Sommet(v)` qui renvoie un ABR réduit à une feuille de valeur  $v$  et de compte de sommets à 1.
- `GreffierSAG(Pere,Fils)` et `GreffierSAD(Pere,Fils)`  
qui greffent l'ABR  $Fils$  respectivement à gauche et à droite de l'ABR  $Pere$

#### Question 2 3 points

Écrire l'algorithme `InsererValeur (v, A)` qui insère la valeur  $v$  dans l'ABR  $A$ .  
On supposera que la valeur  $v$  est absente de  $A$ .

La difficulté de cette question est de maintenir en **chaque** sommet le bon compte du nombre de sommets de l'ABR qui y est enraciné.

En effet, `GreffierSAG` et `GreffierSAD` ne modifient pas ce compte.

On vous demande d'écrire l' algorithme avec une complexité dans  $\theta(h)$ .

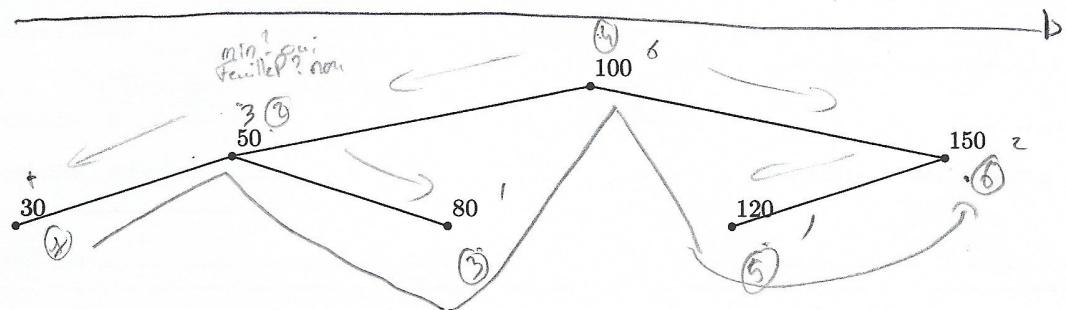
## Utilisation de l'ABR amélioré

On dispose maintenant de la primitive<sup>1</sup> `NbreSomPrim(A)` qui renvoie le compte du nombre de sommets de l'ABR  $A$  (et 0 si  $A$  est vide).

### Question 3 3 points

Soit  $v$  une valeur qui apparaît dans  $A$ .

On veut trouver quel est le rang de  $v$  dans  $A$  (en supposant les valeurs de  $A$  classées par ordre croissant)



Le rang de 120 dans l'ABR ci dessus est 5 : c'est le 5<sup>e</sup> nombre dans la liste 30 50 80 100 120 150 (liste des nombres de l'ABR classée par ordre croissant).

Écrire l'algorithme correspondant<sup>2</sup> `Rang(A, v)` de complexité dans  $\theta(h)$

*Indication :* vous pourrez définir et utiliser un algorithme récursif `RangRec(A, v, k)`.  $k$  sera le rang de la plus petite valeur de  $A$  dans l'arborescence initiale  $A_0$  et `Rang(A0, v)` consistera alors dans l'appel `RangRec(A0, v, 1)`.

Justifier brièvement la complexité de `Rang(A0, v)`.

## Complications

### Question 4 2 points

Que devient la complexité de votre algorithme `Rang` quand vous ne pouvez plus utiliser `NbreSomPrim` mais que vous devez recalculer à chaque fois le nombre de sommets ? (avec la fonction `NbreSommet` que vous avez écrite au début de ce problème) ?

---

1. de complexité dans  $\theta(1)$   
2. qui renvoie ce rang de  $v$



Seconde Session, Licence 2  
Date : 24 Juin 2016

Documents et Matériels autorisés : Aucun  
Durée de l'épreuve : 2 heures

Mention : Informatique  
Algorithmique et Complexité (HLIN401)

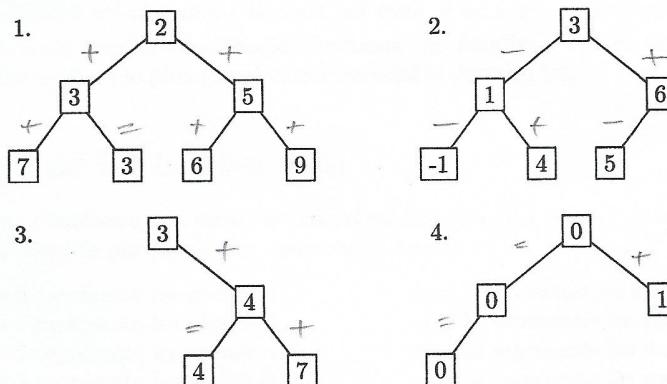
**Consignes générales :** le sujet se compose de deux parties indépendantes. Il est permis de rédiger les deux problèmes sur une même copie. Le langage de programmation naturel pour cet examen est le C++, mais n'est pas obligatoire, à condition que l'étudiant rédige dans un (méta)langage précis et consistant. Il est toujours possible de faire appel à un algorithme auxiliaire s'il est préalablement défini dans la copie.

## Partie 1 : Reconnaissance d'Arbres Binaires Structurés (12pts)

Dans cette section, les sommets d'un arbre sont étiquetés par des entiers quelconques. Deux sommets peuvent avoir la même étiquette. Les définitions d'arbre parfait, de tas et d'arbre binaire de recherche sont celles du cours et ne sont donc pas redonnées. Toutefois, nous rappelons que les tas considérés sont de type *min heap*, c'est à dire que la clef minimale se trouve à la racine.

### Analyse d'exemples (5pts)

**Question 1 (2pts).** Déterminer lesquels des quatre arbres binaires suivants sont des arbres parfaits, des tas, ou des arbres binaires de recherche (attention, un même arbre peut appartenir à plusieurs classes).

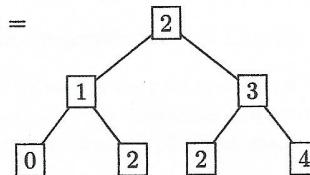


Soit  $x$  un entier et  $h$  un entier positif, on définit l'arbre  $\text{Tree}(x, h)$  par récurrence sur  $h$  comme suit:  $\text{Tree}(x, 0)$  est une feuille dont l'étiquette a valeur  $x$ .  $\text{Tree}(x, h)$  est un arbre de hauteur  $h$  ayant pour racine  $x$ , pour sous arbre gauche  $\text{Tree}(x-1, h-1)$  et pour sous arbre droit  $\text{Tree}(x+1, h-1)$ . Par exemple:

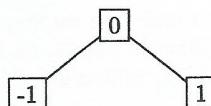
$$\text{Tree}(1, 0) =$$

$$1$$

$$\text{Tree}(2, 2) =$$



$$\text{Tree}(0, 1) =$$



**Question 2** (2pts). Existe-t-il des valeurs de  $x$  et  $h$  telles que  $\text{Tree}(x, h)$  ne soit pas :

1. un arbre parfait ?
2. un tas ?
3. un arbre binaire de recherche ? (justifier)

Indication: lorsque de telles valeurs existent, il est suffisant de donner un exemple pour se justifier, sinon il faut expliquer pourquoi il est impossible d'en trouver.

**Question 3** (1pt). Le nombre de sommets de  $\text{Tree}(x, h)$  dépend-il de  $x$ , de  $h$  ou des deux variables ? Calculer le nombre de sommets de  $\text{Tree}(x, h)$  en fonction des paramètres choisis.

### Algorithmes de reconnaissance (7pts)

On définit l'indice d'un sommet  $s$ , noté  $\text{Index}(s)$ , par induction structurelle sur la hauteur d'un arbre : si  $s$  est une racine,  $\text{Index}(s) = 1$ , sinon si  $s$  est le fils gauche de  $r$ ,  $\text{Index}(s) = 2 \times \text{Index}(r)$ , et sinon, quand  $s$  est le fils droit de  $r$ ,  $\text{Index}(s) = 2 \times \text{Index}(r) + 1$ . L'indice maximum d'un arbre binaire  $T$ , dénoté  $\text{Index}(T)$ , est l'indice le plus grand de ses sommets. Par convention, un arbre vide a pour indice maximum 0. Enfin, le nombre de sommets d'un arbre  $T$  est dénoté par  $\text{Size}(T)$ .

**Question 4** (1pt). Ecrire un algorithme qui calcule l'indice maximum d'un arbre binaire.

**Question 5** (1pt). Justifier qu'un arbre binaire  $T$  est parfait si et seulement si  $\text{Index}(T) = \text{Size}(T)$ .

Indication: observer les indices des sommets dans un parcours en largeur gauche-droite.

**Question 6** (1pt). En déduire un algorithme qui teste si un arbre binaire est parfait.

**Question 7** (2pts). Ecrire un algorithme qui teste si un arbre binaire est un tas. Préciser sa complexité.

**Question 8** (2pts). Ecrire un algorithme linéaire qui teste si un arbre binaire est de recherche.

Indication: si besoin, il est possible d'utiliser les fonctions "int IntMin()" et "int IntMax()" qui renvoient respectivement le plus petit et le plus grand entier encodable dans un int.

### Partie 2 : Jeu de 52 Cartes (8pts)

Dans cette section, on s'intéresse à la structure  $\text{carte}\{\text{int enseigne, int valeur}\}$ . Une carte d'enseigne  $e$  et de valeur  $v$  est aussi dénotée par  $(e, v)$ . Par convention, on a :

$e = 0$ représente les coeurs $\heartsuit$ ,	$v = 1$ représente les as,
$e = 1$ représente les piques $\spadesuit$ ,	$v = 11$ représente les valets,
$e = 2$ représente les carreaux $\diamondsuit$ ,	$v = 12$ représente les dames,
$e = 3$ représente les trèfles $\clubsuit$ .	$v = 13$ représente les rois.
$1 < v \leq 10$ représente les autres valeurs possibles.	

Nous disons qu'une carte est noire si son enseigne est pique  $\spadesuit$  ou trèfle  $\clubsuit$ , et nous disons qu'une carte est rouge si elle est cœur  $\heartsuit$  ou carreau  $\diamondsuit$ . Par exemple, la carte  $(2, 1)$  représente l'as de carreau qui est une carte rouge et l'ensemble  $[(3, 12); (1, 12)]$  est la paire de dames noires.

**Question 9** (1pt). Indiquer quelles sont les cartes présentes dans l'ensemble  $[(0, 3); (3, 1); (1, 13); (2, 4)]$ .

**Question 10** (2pts). Ecrire un algorithme optimal qui détermine si toutes les cartes d'un ensemble sont de la même enseigne, et un second algorithme optimal qui détermine si un ensemble contient au moins une carte de chaque couleur. Justifier que leur complexité est optimale dans le pire des cas.

On souhaite trier des cartes dans l'ordre lexicographique, c'est à dire en regroupant d'abord les enseignes, puis en triant les enseignes par valeur croissante. Formellement, la carte  $(e_1, v_1)$  se situe avant la carte  $(e_2, v_2)$  dans ce tri si et seulement si  $e_1 < e_2$  ou  $e_1 = e_2$  et  $v_1 \leq v_2$ . Ainsi, l'as de pique (1, 1) se situe avant la dame de pique (1, 12) qui se trouve avant le trois de trèfle (3, 3).

**Question 11** (2pts). Sachant qu'il y a un nombre constant de cartes différentes, donner un algorithme linéaire pour trier un ensemble de carte. (indication: utiliser le tri panier, ou tri par comptage)

**Question 12** (1pt). Pourquoi est-il possible d'implémenter un algorithme qui trie un paquet de cartes contenant exactement un exemplaire de chaque carte en temps constant ?  $\Theta(1^n)$

**Question 13** (2pts). Si l'on suppose qu'il n'y a plus de limite sur le nombre d'enseignes ou sur le nombre de valeurs, que devient la complexité d'un tri de cartes par ordre lexicographique ?