NOM : Août 2022

PRÉNOM :

BLOC :

Haute École Léonard de Vinci
Sciences et Techniques
Informatique de gestion *Paul Lambin*

Examen de Mathématiques 2 :

1ère année Bachelier en Informatique de Gestion

BINV1100 – Mathématiques 2

Date : 17 août 2022

Durée de l'examen : 3 heures

Nombre de questions : 6

1. **Sauf avis contraire, toute réponse doit être justifiée.**
2. Si vous n'écrivez pas proprement et lisiblement, votre réponse recevra un zéro.
3. Écrire au crayon est autorisé si le point 2 ci-dessus est respecté.
4. Vous pouvez avoir à votre disposition 10 feuilles recto/verso respectant les conditions suivantes : vos nom et prénom doivent être indiqués, les feuilles doivent être manuscrites, reliées sur toute la longueur de manière à ne pas pouvoir en détacher sans l'arracher et le contenu ne fait pas l'objet de miniaturisation.
5. Pour les questions sur machine, vous devez travailler **sur le U :** . En effet, si vous travaillez ailleurs vos fichiers seront perdus.
6. Les points communiqués en regard des questions sont indicatifs. Des lacunes graves entraîneront l'échec au présent examen.
7. **Mettez vos noms et prénoms au début de chaque question !**

Question 1	**/10**
Question 2	**/10**
Question 3	**/15**
Question 4	**/20**
Question 5	**/10**
Question 6	**/15**
TOTAL	**/80**

Nom : Prénom :

PARTIE I : SUR PAPIER

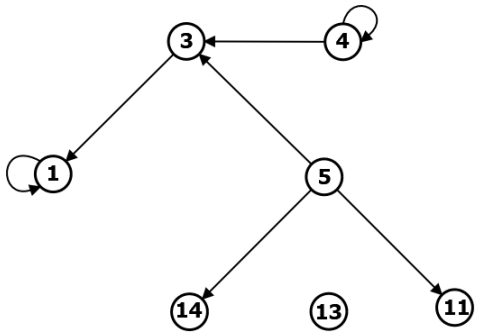
Nom : Prénom :

Question 1 (10 pts)

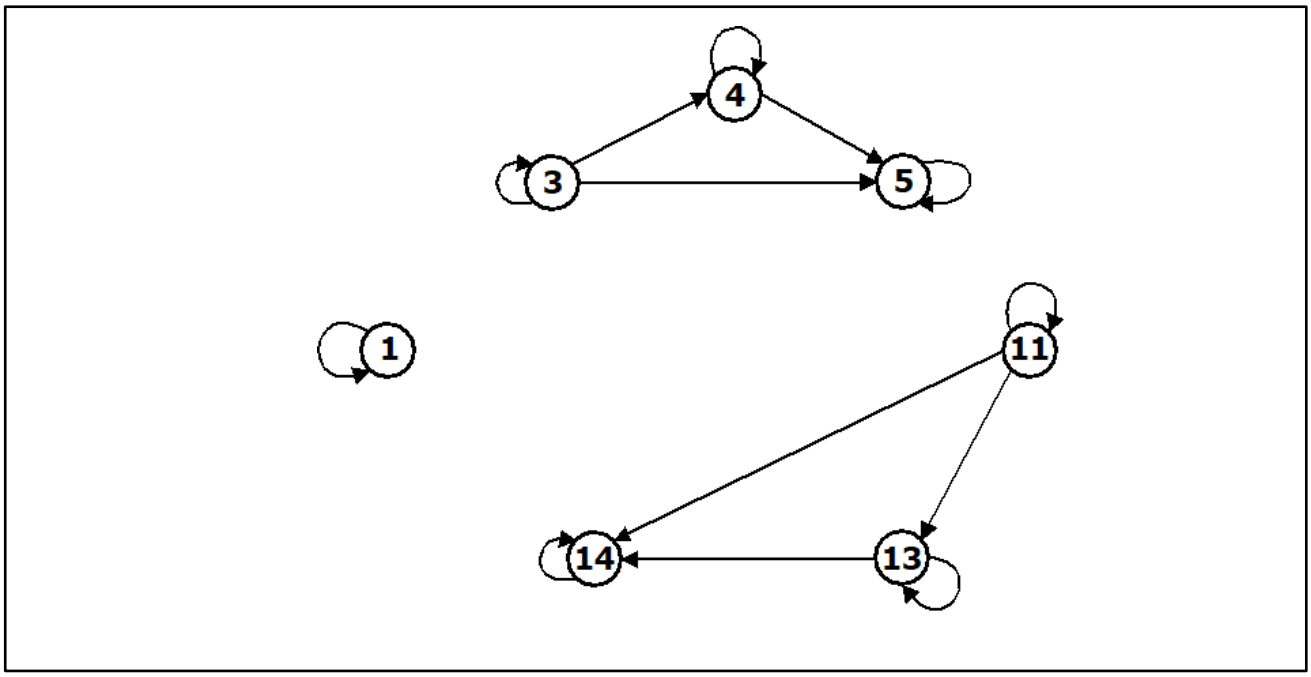
Sur l'ensemble $A = \{1, 3, 4, 5, 11, 13, 14\}$, on considère la relation R définie par

$$xRy \iff x \leq y < 2x$$

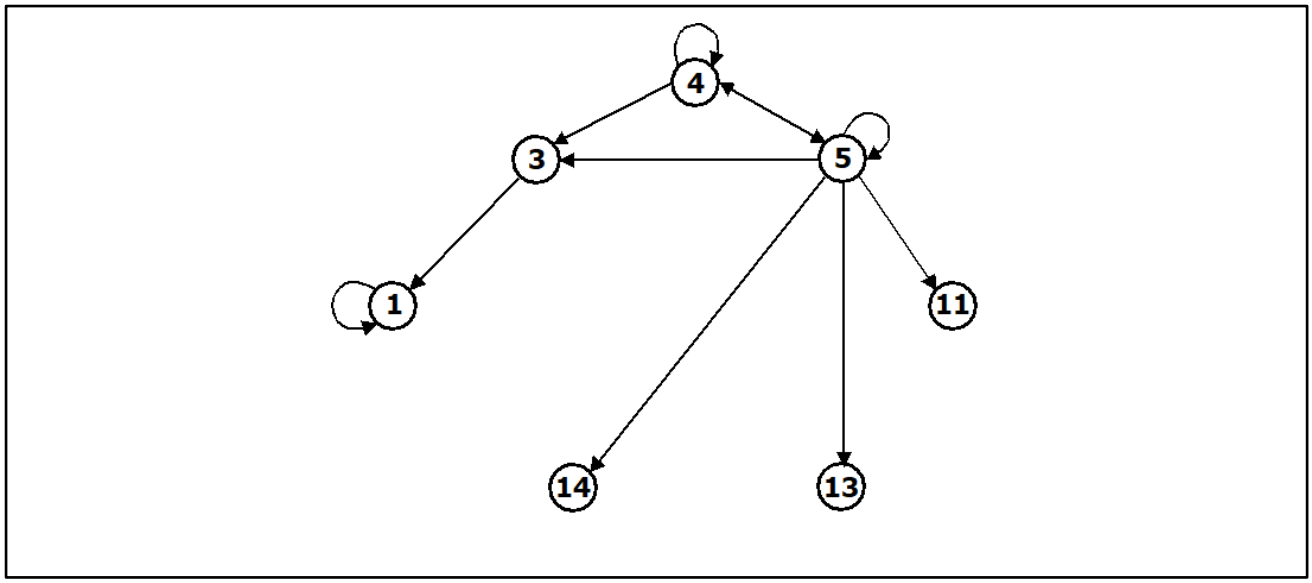
Et la relation S dont voici le digraphe :



a) Donnez le digraphe de R.



b) Donnez le digraphe de $R \circ S$



Nom : Prénom :

c) La relation R est-elle réflexive, antiréflexive, symétrique, antisymétrique, transitive ? **Justifiez pour chaque propriété !!!**

- **Réflexive : la relation R est réflexive car il y a une boucle sur chaque élément.**
- **Antiréflexive : la relation R n'est pas antiréflexive car il y a une boucle sur l'élément 1.**
- **Symétrique : la relation R n'est pas symétrique car il y 3→4 mais pas 4→3**
- **Antisymétrique : la relation R est antisymétrique car il n'y a pas de flèche simple (que des flèches doubles et des boucles).**
- **Transitive : la relation R est transitive car les seuls chemins de longueur de 2 sont 3→4→5 et 11→13→ 14 pour lesquels il y a les chemins de longueur 3→5 et 11→14.**

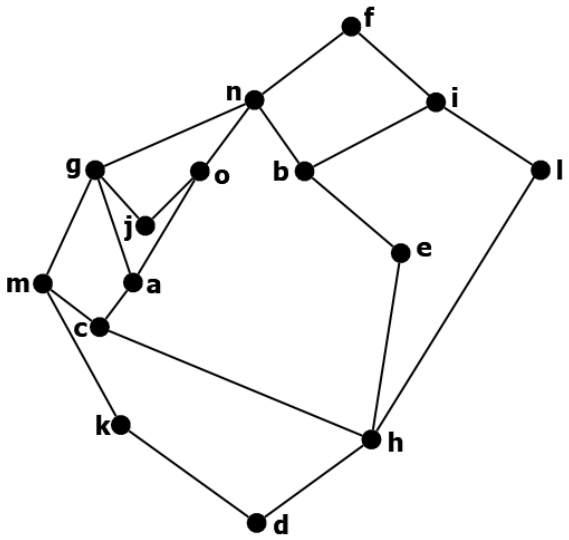
d) Si la relation R est maintenant définie sur N (l'ensemble des naturels), quelle(s) propriété(s) est (sont) conservée(s) ? **Justifiez.**

- **La relation R n'est plus réflexive car (0,0) n'appartient pas à R car $0 \leq 0 < 2 \cdot 0 = 0$ est faux.**
- **La relation R est toujours antisymétrique. En effet si $x \leq y < 2x$ et $x \neq y$ alors $(x, y) \in R$ Mais $y \leq x < 2y$ est faux car si $y \leq x$ alors comme $x \leq y$ on alors $x =$**
  ce qui n'est pas possible car on a supposé $x \neq y$. Donc si $(x, y) \in R$ et $x \neq y$, alors si $(x, y) \notin R$. Et donc R est bien antisymétrique.
- **La relation R n'est plus transitive car $(3, 4) \in R$ car $3 \leq 4 < 2 \cdot 3 = 6$, $(4, 6) \in R$ car $4 \leq 6 < 2 \cdot 8 = 6$. Mais $(3, 6) \notin R$ car $3 \leq 6 < 2 \cdot 3 = 6$ est faux.**

Nom : Prénom :

Question 2 (10 pts)

Soit un ordre partiel $\leq$ sur l'ensemble $E = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o\}$, défini par lede Hasse ci-dessous :



a) Donnez l'ensemble des maximaux de E.

$$\boldsymbol{maximaux(E) = \{f\}}$$

b) L'ordre $\leq$ est-il un treillis ? **Justifiez.**

Non car il y a deux minimaux d et j.

c) Donnez l'ensemble des minimaux de E.

$$\boldsymbol{minimaux(E) = \{d, j\}}$$

d) Donnez l'ensemble des majorants de l'ensemble $B = \{c, e, k\}$

$$\boldsymbol{majorant(B) = \{n, f\}}$$

e) Donnez l'ensemble des minorants de l'ensemble $B = \{c, e, k\}$

$$\boldsymbol{minorant(B) = \{d\}}$$

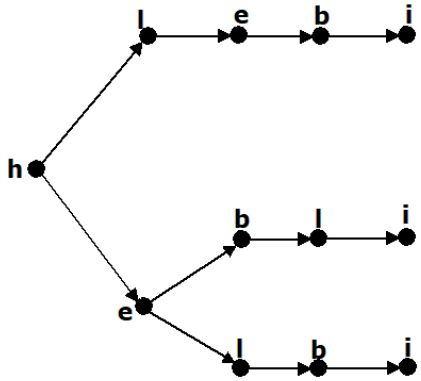
f) Donnez, s'il existe, le supremum de l'ensemble $B = \{c, e, k\}$

$$\boldsymbol{sup(B) = n}$$

Nom : Prénom :

g) Soit l'ensemble $A = \{b, e, h, i, l\}$ ordonné par $\leq$. Combien y a-t-il de tri topologique de A ? **Justifiez.**

Voici l'arbre des tris possibles :



Il y a donc 3 tris topologiques possibles de A.

h) Proposez un sous-ensemble de E, pas totalement ordonné, qui contienne minimum 4 éléments et qui soit un treillis pour l'ordre partiel $\leq$. **Justifiez**

L'ensemble A convient. En effet il a 4 éléments et les seuls couples d'éléments non comparables sont

- **$\{b, l\}$ avec $\inf(\{b, l\}) = h$ et $\sup(\{b, l\}) = i$.**
- **$\{e, l\}$ avec $\inf(\{e, l\}) = h$ et $\sup(\{e, l\}) = i$.**

Donc A est bien un treillis.

Nom : Prénom :

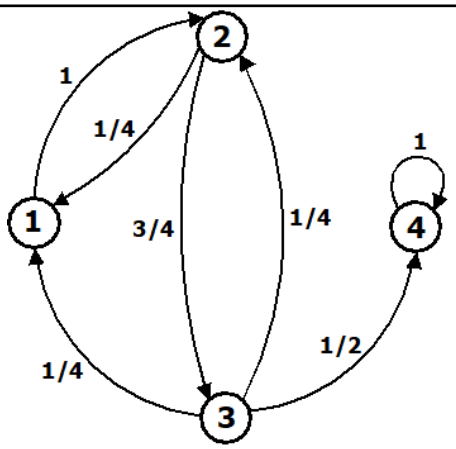
Question 3 (15 pts)

Un étudiant étudie un cours pour la seconde session. Ce cours comporte 3 chapitres qui nécessitent 2h de lecture chacun. Dès qu'il a fini un chapitre, il passe au suivant. Mais

- Après avoir lu le chapitre 1, il passe au chapitre 2 et ne passe jamais du chapitre 1 au chapitre 2.
- Après avoir lu le chapitre 2, il a une chance sur 4 d'avoir besoin d'une notion du chapitre 1 qu'il n'a pas bien assimilée donc de devoir retourner lire le chapitre 1 en entier. Sinon il passe au chapitre 3.
- Après avoir lu le chapitre 3, il a une chance sur 4 d'aller relire le chapitre 1 et une chance sur 4 d'aller relire chapitre 2 pour revoir des notions mal comprises. Sinon il considère qu'il a terminé l'étude de ce cours.
- Si l'étudiant considère qu'il a terminé l'étude du cours alors il arrête définitivement sa lecture.

On vous demande de modéliser cette situation sous la forme d'un processus de Markov, l'unité de temps étant le temps nécessaire à la lecture d'un chapitre.

1. Dessinez le graphe du processus de Markov associé.



État 1 : Chapitre 1

État 2 : Chapitre 2

État 3 : Chapitre 3

État 4 : Étude terminée

2. Donnez la matrice de transition du processus

$$P = \begin{pmatrix} 0 & 1/4 & 1/4 & 0 \\ 1 & 0 & 1/4 & 0 \\ 0 & 3/4 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \end{pmatrix}$$

Nom : Prénom :

3. Donnez les classes de communication, la période de chaqute classe et la nature de chaque état.

$C_1 = \{1, 2, 3\}$, transients, période = PGCD(2,3,…) = 1

$C_2 = \{4\}$, récurrent, période = 1.

4. Le processus admet-il une distribution stable ? **Justifiez**. Si oui, Est-elle unique ? **Justifiez sans la calculer !**

Le processus admet une distribution stable car tout processus de Markov en a une. Par contre on ne sait pas si elle est unique car le processus n'est pas irréductible puisqu'il a 2 classes de communication.

5. On désire déterminer le temps moyen qu'il faudra à l'étudiant pour étudier l'ensemble du cours. Donnez le système, la matrice à échelonner ainsi que la variable à déterminer pour trouver ce temps de parcours. **Ne résolvez pas le système et n'échelonnez pas sa matrice !**

On cherche le temps moyen pour aller de l'état 1 à l'état 4.

Soit t_i le temps pour aller de l'état i à l'état 4. Alors on a le système suivant

$$\begin{cases} t_1 = 1 + t_2 \\ t_2 = 1 + \frac{1}{4}t_1 + \frac{3}{4}t_3 \\ t_3 = 1 + \frac{1}{4}t_1 + \frac{1}{4}t_2 + \frac{1}{2}t_4 \\ t_4 = 0 \end{cases} \Rightarrow \begin{cases} t_1 - t_2 = 1 \\ -\frac{1}{4}t_1 + t_2 - \frac{3}{4}t_3 = 1 \\ -\frac{1}{4}t_1 - \frac{1}{4}t_2 + t_3 = 1 \end{cases}$$

Donc la matrice à échelonner est donc $\left(\begin{array}{ccc|c} 1 & -1 & 0 & 1 \\ -1/4 & 1 & -3/4 & 1 \\ -1/4 & -1/4 & 1 & 1 \end{array}\right)$

Et la variable à déterminer est t_1.

Nom : Prénom :

Question 4 (20 pts)

1) a) Soit L le langage, sur l'alphabet $\Sigma = \{a, b, c\}$, formé de tous les mots de minimum 2 lettres comprenant au maximum 1 b et dans lesquels un c est toujours suivi d'un a. Donnez une grammaire régulière engendrant ce langage. Précisez bien à quel « état » correspond chaque symbole non terminal !

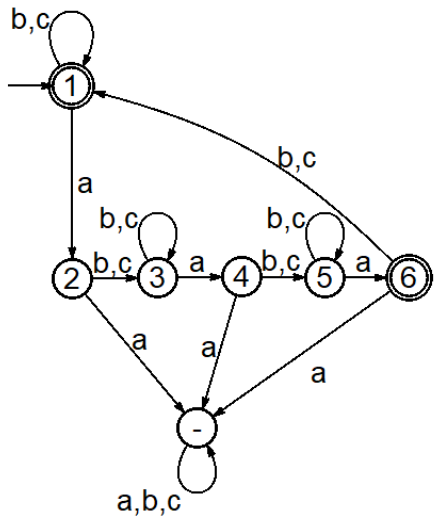
<S> : 0 lettre
<A> : une lettre a
<B> : une lettre b
<C> : termine par c et contient un seul b
<D> : termine par c et ne contient pas de b
<E> : ne termine pas par c et contient un seul b
<F> : ne termine pas par c et ne contient pas de b

Grammaire :

<S> → a<A> | b<B> | c<D>
<A> → a<F> | b<E> | c<D>
<B> →a<E> | c<C>
<C> → a<E>
<D> → a<F>
<E> → Ɛ |a<E> | c<C>
<F> → Ɛ | a<F> | b<E> | c<D>

b) Soit L le langage, sur l'alphabet $\Sigma = \{a, b, c\}$, formé de tous les mots où le nombre de a est divisible par 3 mais où il n'y a jamais deux a consécutifs. Donnez un automate de Moore reconnaissant ce langage. Précisez bien à quel mot correspond chaque état !

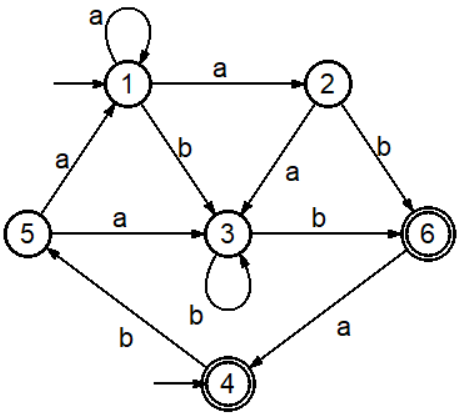
État 1 : contient 3n fois a et ne termine pas par a
État 2 : contient 3n+1 fois a et termine par a
État 3 : contient 3n+1 fois a et ne termine pas par a
État 4 : contient 3n+2 fois a et termine par a
État 5 : contient 3n+2 fois a et ne termine pas par a
État 6 : contient 3n fois a et termine par a



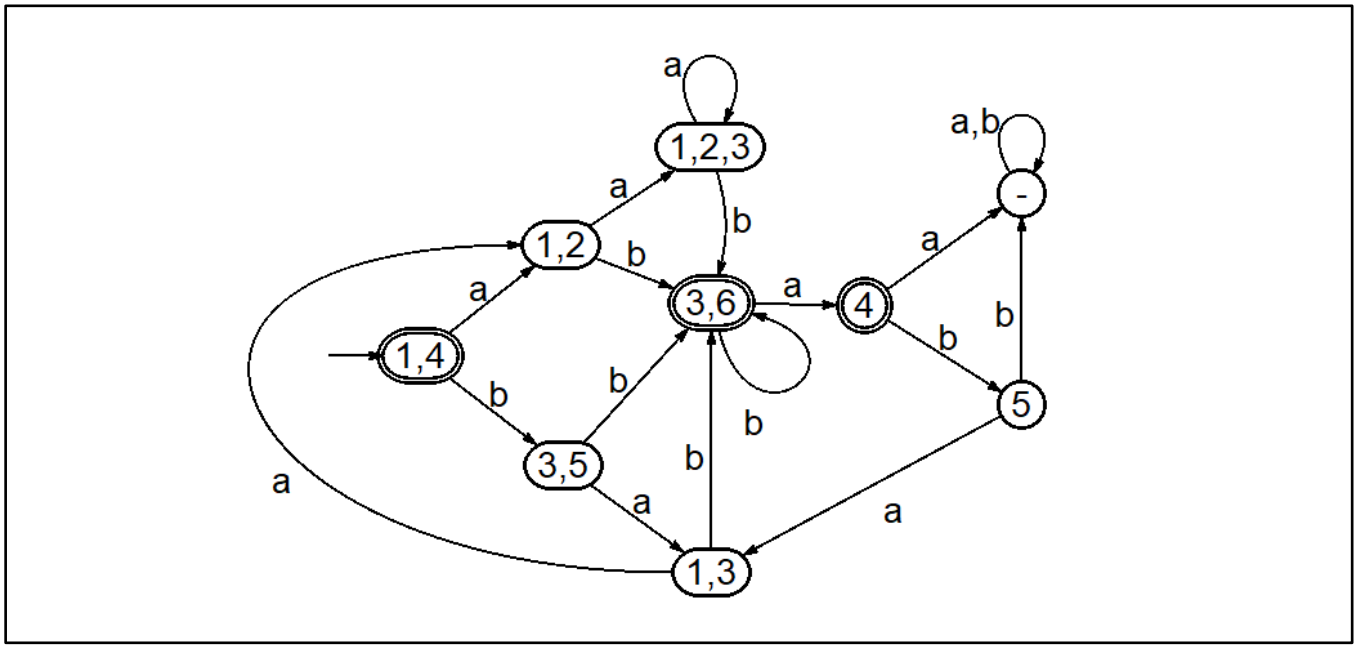
Nom : Prénom :

2) Soit L le langage défini sur $\Sigma = \{a, b\}$ reconnu par le NDFA M.

M



Utilisez la subset construction afin d'obtenir un automate de Moore reconnaissant le langage L :

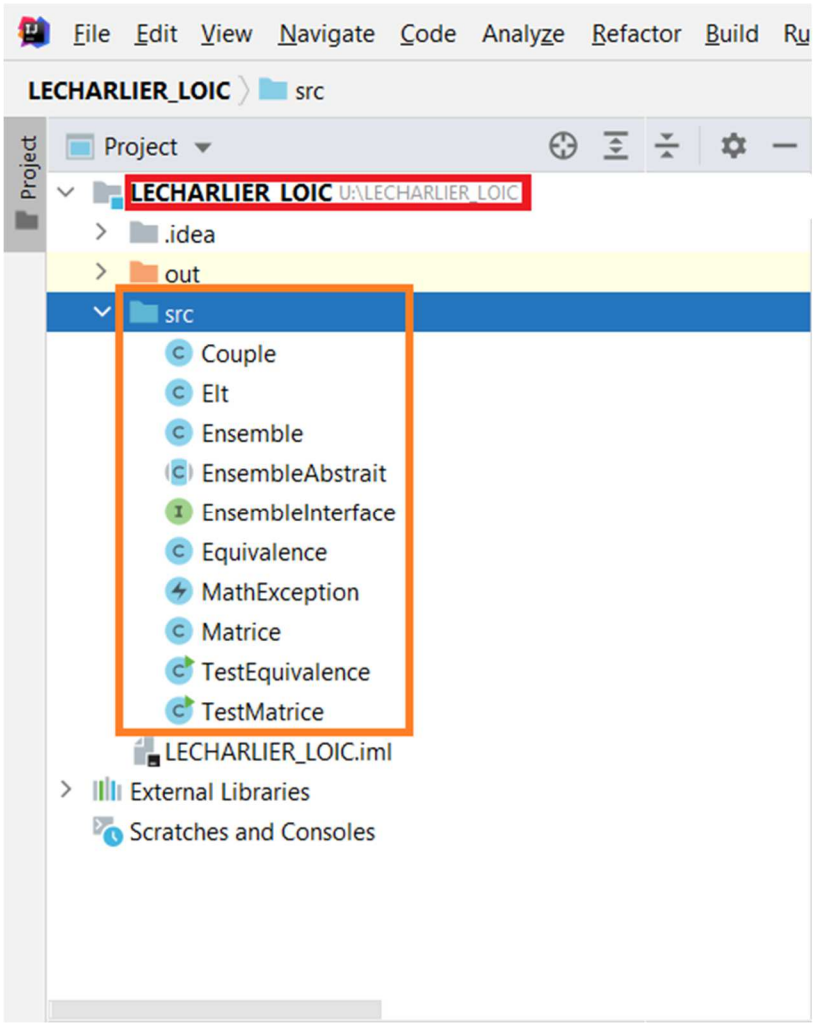


Nom : Prénom :

PARTIE II : IMPLÉMENTATION

Dans cette partie nous allons vous demandez d'implémenter des méthodes dans plusieurs classes classes Java.
Pour ce faire :

1) Ouvrez IntelliJ
2) Créez, **sur le U :**, un projet NOM_PRENOM (**avec vos nom et prénom** !)
3) Les classes données se trouvent dans le répertoire « Classes Java ». Faites un copier-coller de celles-ci dans le répertoire « src » de votre projet IntelliJ. Voici ce que vous devriez obtenir :



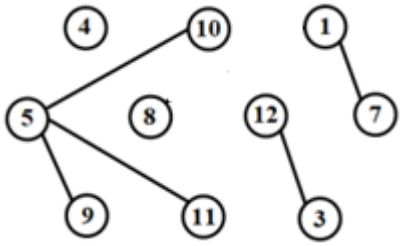
Les questions 5 et 6 ci-après vous expliquerons ce que vous devez implémenter.

Question 5 (10 pts)

Nom : Prénom :

Les instances de la classe *Equivalence* sont des relations d'équivalence sur une partie de l'univers. La partie de l'univers sur laquelle est définie la relation d'équivalence est stockée dans un *EnsembleAbstrait* (sousJac).
On a choisi de représenter une équivalence en choisissant pour chaque classe un représentant. Afin de réaliser cela, la classe Equivalence garde un tableau d'*Elt* (tabRep) dans lequel on stocke à l'indice correspondant à l'*Elt* le représentant de sa classe.

Exemple :
Soit ~ une relation d'équivalence sur {1,3,4,5,7,8,9,10,11,12}.



Cette relation ~ a 5 classes d'équivalence : {4}, {5,9,10,11}, {1,7}, {8} et {3,12}
On choisit un représentant par classe.
Par exemple : {**4**}, {5,**9**,10,11}, {1,**7**}, {**8**} et {**3**,12}

sousJac = {1,3,4,5,7,8,9,10,11,12}
tabRep =

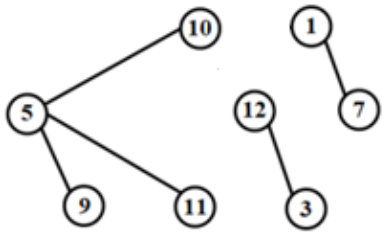
\	7	\	3	4	9	\	7	8	9	9	9	3	\		\
0	1	2	3	4	5	6	7	8	9	10	11	12	13	…	MAX

Définition : L'ensemble des isolés est l'ensemble des éléments du sous-jacent qui ne sont en relation avec aucun autre élément.

Dans l'exemple : il y a 2 éléments isolés : 4 et 8.

Dans la classe *Equivalence*, complétez la méthode supprimerLesIsoles().
Testez-la avec la classe *TestEquivalence*.

Après l'exécution de la méthode supprimerLesIsoles, la relation d'équivalence précédente devient



Avec

sousJac = {1,3,5,7,9,10,11,12}
tabRep =

\	7	\	3	\	9	\	7	\	9	9	9	3	\		\
0	1	2	3	4	5	6	7	8	9	10	11	12	13	…	MAX

Attention ! Cette classe ***Equivalence*** n'implémente pas *RelationAbstraite* ! Elle n'a donc ni de méthode contient() ni d'itérateur sur les couples de la relation.
Par contre, la classe *Ensemble* fournie implémente *Iterable*.

```java
public class Equivalence{

    /** Valeur numerique de MAXELT */
    private static final int MAX = Elt.MAXELT.val();

    private EnsembleAbstrait sousJac; // ensemble sous-jacent
    private Elt[] tabRep; // tableau des representants
    // Ne pas ajouter d'attribut

    //modifie l'equivalence courante en supprimant tous les elements isoles
    public void supprimerLesIsoles(){
        //TODO
      EnsembleAbstrait sJ = sousJac.clone() ;
      for (Elt e:sJ) {
         if (tabRep[e.val()].equals(e)) {
            boolean isole = true ;
            for (Elt f:sousJac) {
               if (tabRep[f.val()].equals(e) && !f.equals(e)) {
                  isole = false ;
                  break ;
               }
            }
            if (isole) {
               sousJac.enlever(e) ;
               tabRep[e.val()]=null ;
            }
         }
      }
    }

    //NE PAS MODIFIER : va servir pour les tests
    public Equivalence(Elt[] tabRepARecopier) {
        sousJac = new Ensemble();
        tabRep = new Elt[MAX+1];
        for (int i = 1; i < tabRepARecopier.length; i++) {
            if(tabRepARecopier[i]!=null){
                sousJac.ajouter(new Elt(i));
                tabRep[i]=tabRepARecopier[i];
            }
        }
    }



}

```

Nom : Prénom :

Question 6 (15 pts)

On vous demande d'implémenter la méthode

`echelonner(int numLigne, int[] lignes, double[] coeff)`

de la classe `Matrice`.

Cette méthode va remplacer la ligne `numLigne`, par la somme des lignes contenue dans le tableau `lignes`, chaque ligne étant pondérée par un coefficient se trouvant dans le tableau `coeff`.

Exemple :

Si on veut faire $\ell_3 \longleftarrow \ell_3 - 0.5\,\ell_1 + 1.5\,\ell_2$ alors

- `numLignes = 3` (on veut modifier la ligne 3)
- `lignes =` | 3 | 1 | 2 |
- `coeff =` | 1 | -0.5 | 1.5 |

Mais on veut que la modification respecte les règles de l'échelonnage, il y a donc quelques conditions à vérifier :

- ✓ `nSupumLigne` est un numéro de ligne valide
- ✓ `lignes` et `coeff` ne sont pas null
- ✓ `lignes` et `coeff` doivent avoir la même taille
- ✓ `lignes` contient des numéros de lignes valide
- ✓ `lignes` ne contient pas deux fois le même numéro de ligne
- ✓ `lignes` doit contenir numLigne (on doit garder les infos de la ligne à modifier)
- ✓ Le coefficient contenu dans `coeff` correspondant à `numLigne` ne peut pas être égal à 0 (on doit garder les infos de la ligne à modifier)

Si une de ces conditions n'est pas vérifiées, il faut lancer une `IllegalArgumentException`.

Exemple : Supposons que `this` est une matrice 3×3

1) Si `numLigne` = 5 alors `IllegalArgumentException` car `this` n'a pas de ligne 5.
2) Si `numLigne` = 3 et `lignes =` | 3 | 1 | 1 | alors `IllegalArgumentException` car `lignes` contient deux fois la ligne 1.
3) Si `numLigne` = 3 et `lignes =` | 3 | 1 | -1 | alors `IllegalArgumentException` car `lignes` contient -1 qui est un numéro de ligne qui n'existe pas.
4) Si `numLigne` = **3** et `lignes =` | 1 | 2 | alors `IllegalArgumentException` car `lignes` ne contient pas **3** qui est la ligne que l'on veut modifier.
5) Si `numLigne` = **3**, `lignes =` | 3 | 1 | 2 | et `coeff =` | 0 | 0.5 | -1 | alors `IllegalArgumentException` car le coefficient, contenu dans `coeff`, correspondant à `numLigne` est égal à 0.

```java
import java.util.Arrays ;

public class Matrice {
    private final int nbLignes;              // nombre de lignes
    private final int nbColonnes;            // nombre de colonnes
    private final double[][] data;           // matrice (nbLignes,nbColonnes)

    // ce constructeur cree la matrice nulle de genre (a,b)
    public Matrice(int a, int b) throws IllegalArgumentException {
        if (a<=0 || b<=0)
            throw new IllegalArgumentException("a ou b négatif") ;
        data = new double[a][b] ;
        nbLignes = a ;
        nbColonnes = b ;
    }

    public void echelonner(int numLigne, int[] lignes, double[] coeff) {
        if (numLigne<0 || numLigne>nbLignes)
            throw new IllegalArgumentException("Numéro de ligne inexistant") ;
        if (lignes==null)
            throw new IllegalArgumentException("lignes == null") ;
        if (coeff==null)
            throw new IllegalArgumentException("coeff == null") ;
        if (lignes.length!=coeff.length)
            throw new IllegalArgumentException("lignes et coeff pas de la même
            longueur") ;
        boolean contientNumLigne = false ;
        for (int i=0 ; i< lignes.length ; i++) {
            if (lignes[i]<=0 || lignes[i]>nbLignes) {
                throw new IllegalArgumentException("Numéro de ligne inexistant") ;
            }
            if (coeff[i]==0)
                throw new IllegalArgumentException("un coefficient est égal à 0") ;
            if (lignes[i]==numLigne) {
                if (contientNumLigne)
                    throw new IllegalArgumentException("Deux fois numLigne") ;
                contientNumLigne = true ;
            }
        }
        if (!contientNumLigne) {
            throw new IllegalArgumentException("ligne à modifiée pas contenue dans la
            combinaison") ;
        }
        for (int j=0 ; j<this.nbColonnes ; j++) {
            double somme = 0 ;
            for (int k=0 ; k<lignes.length ; k++) {
                somme=somme+this.data[lignes[k]-1][j]*coeff[k] ;
            }
            this.data[numLigne-1][j]=somme ;
        }
    }

    // affiche la matrice en format standard
    public String toString(){
        String st = "" ;
        int tmax = 0 ;
        for (int i=0 ; i<nbLignes ; i++) {
            for (int j=0 ; j<nbColonnes ;j++) {
                String s = "" + data[i][j] ;
                if (data[i][j]>=0)
                    s = " "+s ;
                if (s.length()>tmax)
                    tmax = s.length() ;
            }
        }
        for (int i=0 ; i<nbLignes ; i++) {
            for (int j=0 ; j<nbColonnes ;j++) {
                String s = "" + data[i][j] ;
                if (data[i][j]>=0)
                    s = " "+s ;
                st = st + s ;
                int nbBlanc = tmax-s.length()+2;
                for (int k=0 ; k<nbBlanc ; k++)
```