NOM : Juin 2023

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 : 7 juin 2022

Durée de l'examen : 2 heures et 30 minutes

Nombre de questions : 5

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	**/20**
Question 2	**/20**
Question 3	**/20**
Question 4	**/15**
Question 5	**/15**
TOTAL	**/90**

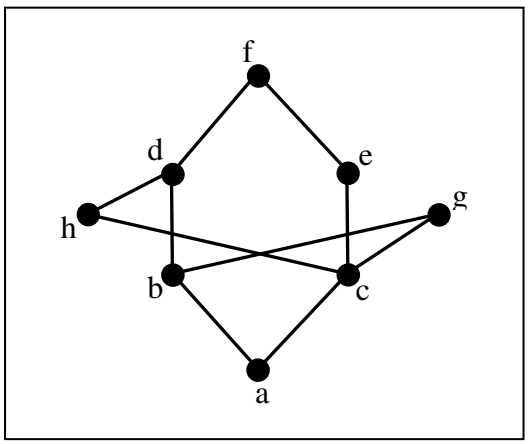
Nom : Prénom :

PARTIE I : SUR PAPIER

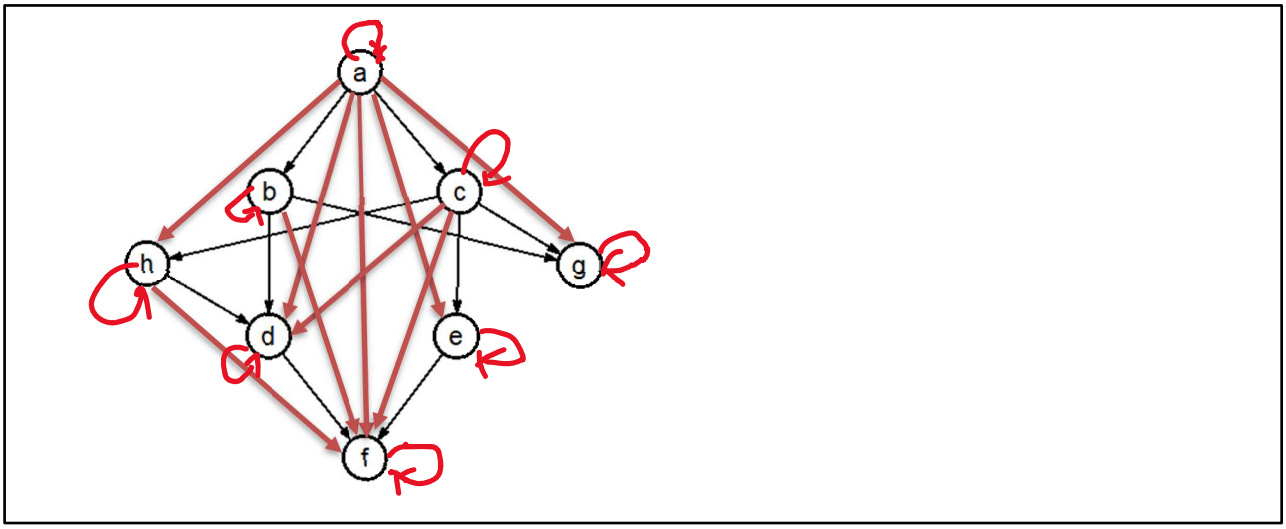
Nom : Prénom :

Question 1 (20 pts)

D Soit un ordre partiel ≤ sur l'ensemble E = {a, b, c, d, e, f, g, h}, défini par le diagramme de Hasse ci-dessous :



a) Complétez le digraphe qui suit pour qu'il représente la même relation :



b) Pour chacun des ordres totaux ci-dessous, indiquez « oui » si c'est un tri topologique de ≤, et « non » sinon. Si c'est « non », justifiez votre assertion.

1 : a → b → c → g → e → h → d → f
2 : a → b → h → c → e → d → g → f
3 : a → c → g → e → b → h → d → f

1 : Oui
2 : Non car c<h. Donc c doit être pris avant h
3 : Non car b<g. Donc b doit être pris avant g

Nom : Prénom :

c) L'ordre partiel ≤ sur E est-il un treillis ? Justifiez votre réponse.

Non car f et g sont deux maximaux non comparable donc la paire {f,g} n'a pas de supremum.

d) Indiquez si ≤ sur E possède un minimum, un maximum, un infimum et un suprémum. Dans l'affirmative, donnez l'élément. Dans la négative, justifiez.

- **Min(E) = a.**
- **E n'a pas de maximum car E a deux maximaux (f et g) non comparables**
- **Inf(E) = a car c'est le min**
- **E n'a pas de supremum car E a deux maximaux non comparables**

e) Relativement à ≤ sur E, déterminez les 4 ensembles suivants : les minimaux, les maximaux, les minorants et les majorants.

- **Minimaux(E) = {a}**
- **Maximaux(E)= {f,g}**
- **Minorant(E)= {a}**
- **Majorant(E)= { }**

f) Proposez un sous-ensemble de E possédant au moins 5 éléments et qui soit un treillis pour ≤.

{a,b,c,d,f} car la seule paire d'élément non comparable est {b,c}

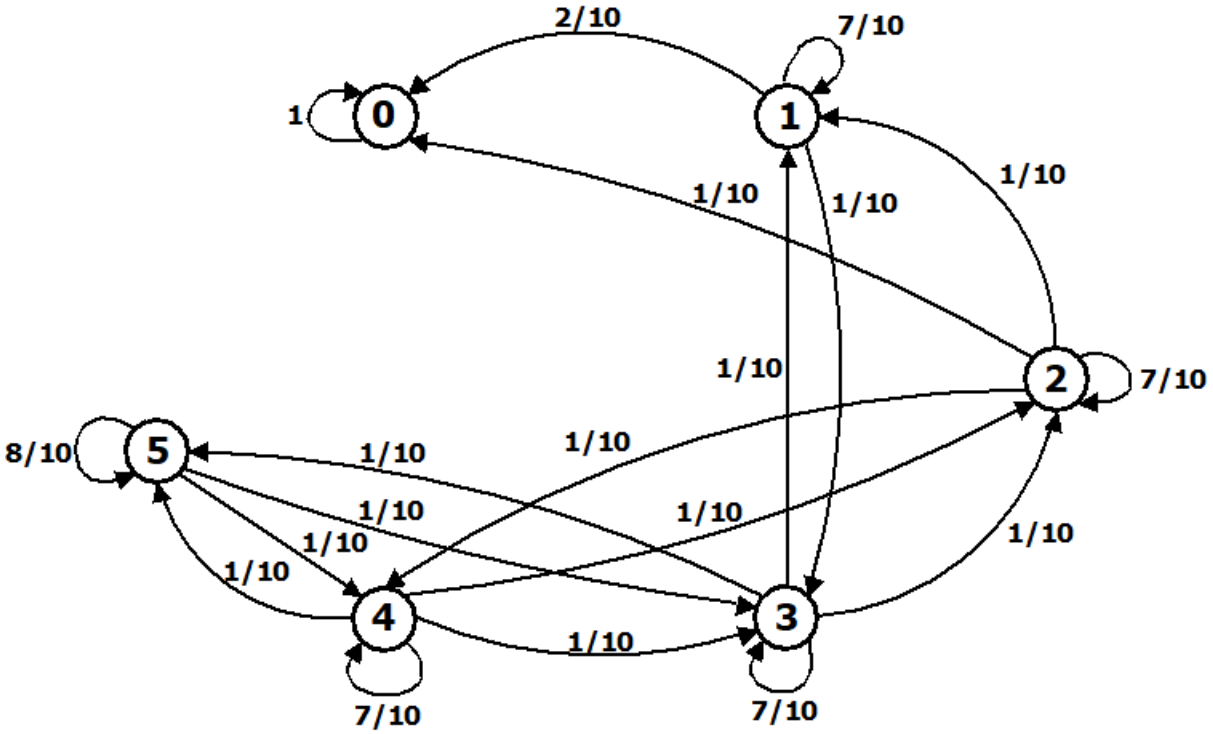
avec inf({b,c}) = a et sup({b,c}) = d

Nom : Prénom :

Question 2 (20 pts)

1) Le Salsyn est un jeu traditionnel qui se joue avec un dé à dix faces. Chaque joueur démarre la manche avec 5 jetons. A son tour, il lance le dé. S'il fait 2, il perd un jeton. S'il fait un 1, il perd deux jetons (sans pouvoir descendre sous 0 jeton). En revanche, s'il fait un 10, il regagne 2 jetons (sans pouvoir grimper au-dessus de 5 jetons). Le jeu se termine quand le joueur n'a plus de jeton.

   a. Complétez le diagramme du processus de Markov et écrivez la matrice de transition.

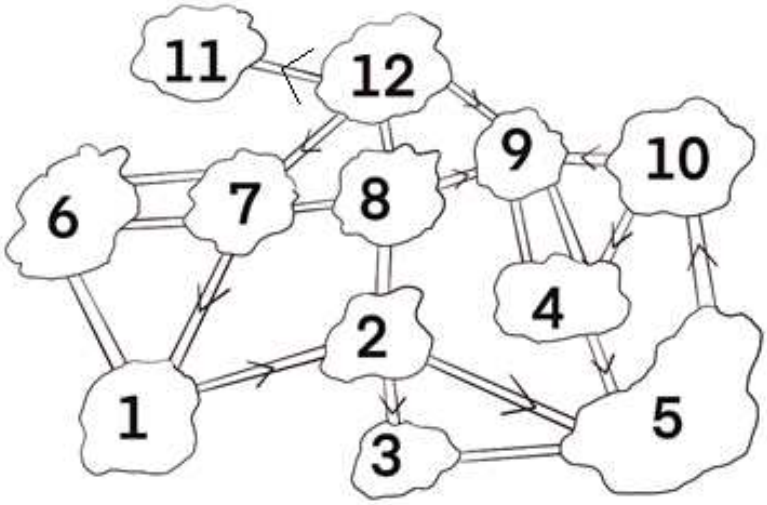


Matrice de transition :

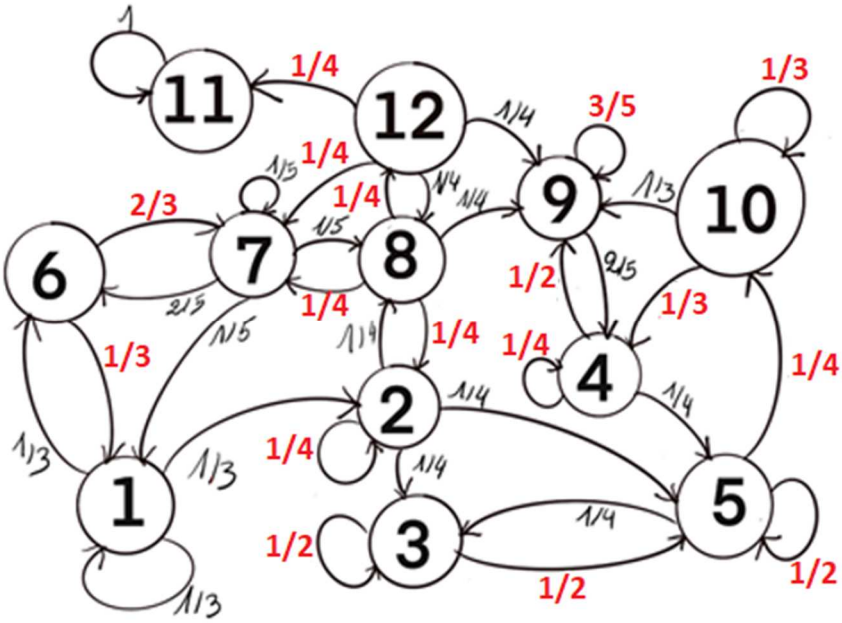
$$P = \begin{pmatrix} 1 & 2/10 & 1/10 & 0 & 0 & 0 \\ 0 & 7/10 & 1/10 & 1/10 & 0 & 0 \\ 0 & 0 & 7/10 & 1/10 & 1/10 & 0 \\ 0 & 1/10 & 0 & 7/10 & 1/10 & 1/10 \\ 0 & 0 & 1/10 & 0 & 7/10 & 1/10 \\ 0 & 0 & 0 & 1/10 & 1/10 & 8/10 \end{pmatrix}$$

Nom : Prénom :

2) Ce schéma correspond à un labyrinthe doté d'un système de portes. Celles marquées d'une flèche ne peuvent être passées que dans le sens de la flèche, les autres pouvant être passées dans les 2 sens. Elles sont symbolisées par une flèche > . Un héros perdu dedans choisira aléatoirement une des portes s'offrant à lui. S'il essaie de passer une porte dans le mauvais sens, il va rester dans la pièce où il est avant de refaire son choix de la même manière.



a. Complétez le diagramme de Markov ci-dessous.



b. Partant du principe que la sortie est dans la salle n°11 et que le héros est dans la salle n°1, est-il possible que le héros se retrouve bloquer sans pouvoir atteindre la sortie ? Si oui, dans quelle(s) salle(s) ? Justifiez en vous servant de la notion de classe de communication.

Il y a 3 classes de communication : $C_1 = \{1, 2, 6, 7, 8, 12\}$, $C_2 = \{3, 4, 5, 9, 10\}$ et $C_3 = \{11\}$. Or les classes C_2 et C_3 sont toutes les 2 récurrentes. Donc si le héros entre les salles 3, 4, 5, 9 ou 10, il restera bloqué dans la classe C_2 sans jamais pouvoir en sortir et il n'atteindra jamais la sortie.

Nom : Prénom :

3) Soit un jeu générant la matrice de Markov ci-dessous. On aimerait connaître le temps moyen pour atteindre l'état 5 en partant de l'état 1. Donnez le système, sans le résoudre, ainsi que l'inconnue à déterminer.

$$\begin{pmatrix} 1/4 & 1/2 & 1/4 & 2/7 & 0 \\ 1/8 & 1/3 & 0 & 2/7 & 1/6 \\ 1/4 & 0 & 1/2 & 1/7 & 0 \\ 0 & 1/6 & 1/8 & 0 & 1/2 \\ 3/8 & 0 & 1/8 & 2/7 & 1/3 \end{pmatrix}$$

Soit t_i le temps de parcours de l'état i à l'état 5. Alors le système à résoudre est le suivant :

$$\begin{cases} t_1 = 1 + \frac{1}{4}t_1 + \frac{1}{8}t_2 + \frac{1}{4}t_3 + \frac{3}{8}t_5 \\ t_2 = 1 + \frac{1}{2}t_1 + \frac{1}{3}t_2 + \frac{1}{6}t_4 \\ t_3 = 1 + \frac{1}{4}t_1 + \frac{1}{2}t_3 + \frac{1}{8}t_4 + \frac{1}{8}t_5 \\ t_4 = 1 + \frac{2}{7}t_1 + \frac{2}{7}t_2 + \frac{1}{7}t_3 + \frac{2}{7}t_5 \\ t_5 = 0 \end{cases}$$

Et la variable à déterminer est t_1.

Nom : Prénom :

Nom : Prénom :

Question 3 (20 pts)

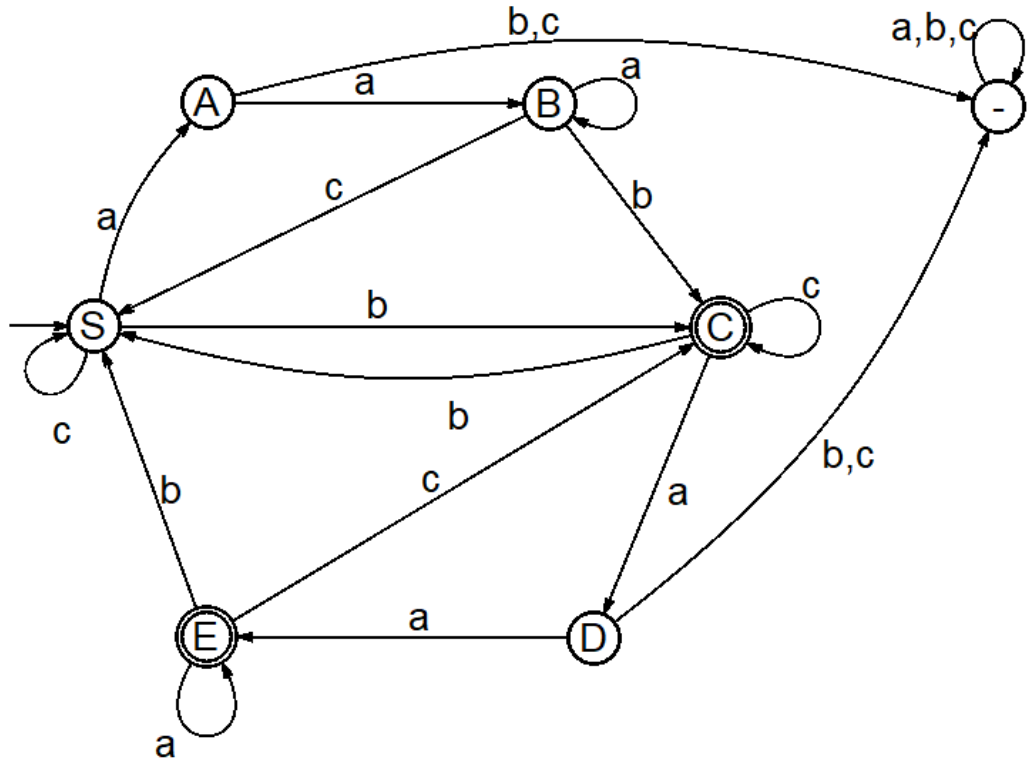
a) Soit L le langage, sur l'alphabet $\Sigma = \{a, b, c\}$, formé de tous les mots qui comportent un nombre impair de b et dans lesquels chaque a est toujours suivi ou précédé par un autre a. Donnez une grammaire régulière engendrant ce langage ou un automate de Moore le reconnaissant. Précisez bien à quel mot correspond chaque symbole non terminal ou chaque état !

<S> : Nombre pair de b et ne finit pas par a
<A> : Nombre pair de b finit par a mais pas par aa
<B> : Nombre pair de b et finit par aa
<C> : Nombre impair de b et ne finit pas par a
<D> : Nombre impair de b et finit par a mais pas par aa
<E> : Nombre impair de b et finit par aa

Grammaire :

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

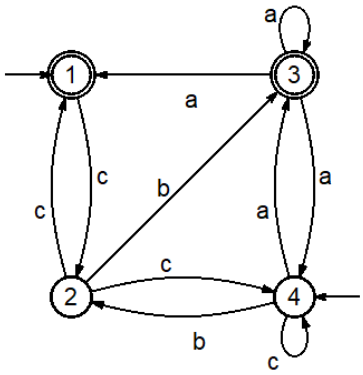
Automate de Moore :



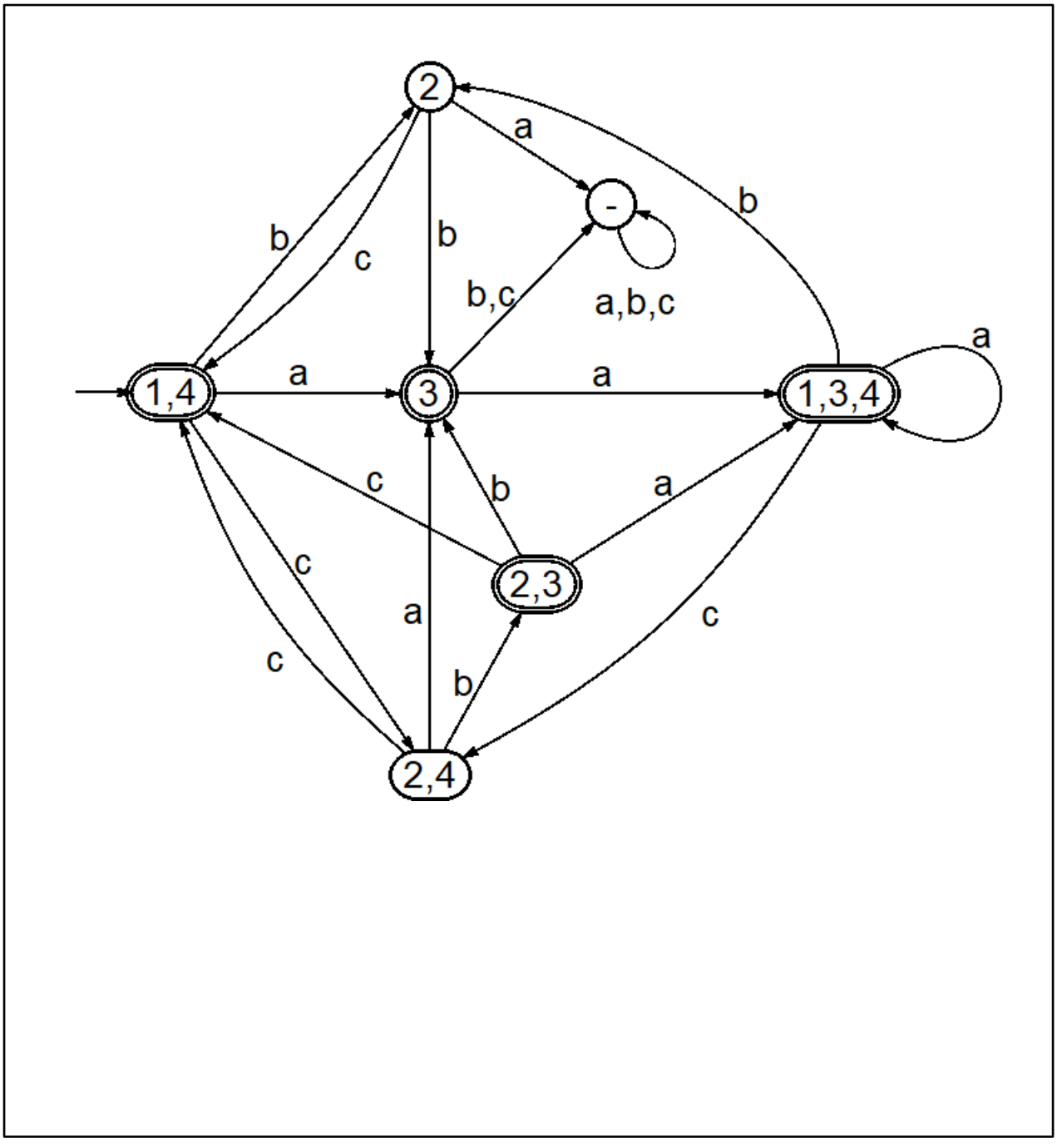
Nom : Prénom :

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

M



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



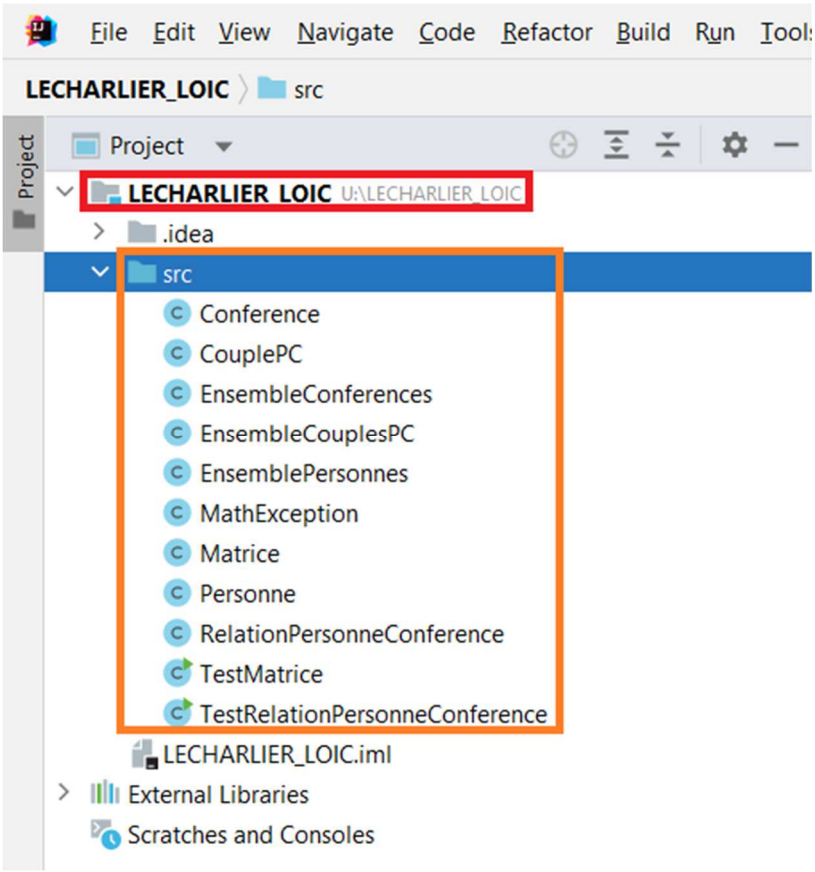
Nom : Prénom :

PARTIE II : IMPLÉMENTATION

Dans cette partie nous allons vous demander d'implémenter des méthodes dans plusieurs 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 :

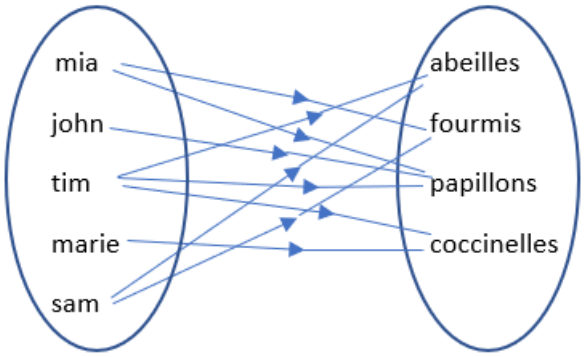


Les questions 4 et 5 ci-après vous expliquerons ce que vous devez implémenter.

Nom : Prénom :

Question 4 (15 pts)

Plusieurs conférences sont proposées pendant toute la durée d'une exposition sur les insectes. Lors de son inscription, chaque personne a donné son nom, mais également une liste des conférences auxquelles elle va assister. Chaque personne s'inscrit à au moins une conférence. Deux personnes vont se croiser si elles se sont inscrites à au moins une même conférence. Nous allons nous intéresser à la relation qui unit une personne à une conférence :



Mia s'est inscrite à la conférence sur les fourmis et à celle sur les papillons.
John s'est inscrit à la conférence sur les papillons.
Tim s'est inscrit à la conférence sur les abeilles, à celle sur les papillons et à celle sur les coccinelles.
…
John et Marie se sont inscrits à 1 conférence.
Mia et Sam se sont inscrits à 2 conférences.
Tim s'est inscrit à 3 conférences.

Mia va croiser John car ils sont tous les 2 inscrits à la conférence sur les papillons.
Elle croisera également Tim et Sam, mais pas Marie.
…

Implémentation :

Voici les classes qui sont fournies (à ne pas modifier) : `Personne`, `Conference`, `CouplePC`, `EnsemblePersonnes`, `EnsembleConference` et `EnsembleCouplesPC`.
Les différents ensembles possèdent les méthodes classiques sur les ensembles.
Ces ensembles peuvent être itérés :

```
for(Personne p : ensemblePersonnes)

for(Conference c : ensembleConferences)

for(CouplePC cpc : ensembleCouplesPC)
```

Vous allez compléter 2 méthodes de la classe `RelationPersonneConference`.
Cette classe possède comme attributs 3 ensembles : EnsemblePersonnes + EnsembleConferences + EnsembleCouplesPC.
Elle possède un constructeur qui instancie ces 3 ensembles.
Elle contient les méthodes `estVide()`, `contient(CouplePC cpc)`, `contient(Personne p, Conference c)`, `ajouter(Personne p, Conference c)` et `ajouter(CouplePC cpc)`.

Les 2 méthodes à compléter sont : `inscritesA2Conferences()` et `vontSeCroiser(Personne p1, Personne p2)`.

Nom : Prénom :

La méthode `inscritesA2Conferences()` renvoie l'ensemble des personnes qui sont inscrites à exactement 2 conférences.

La méthode `vontSeCroiser(Personne p1, Personne p2)` vérifie si les 2 personnes ont au moins une conférence en commun.

La classe `TestRelationPersonneConference` permet de tester les 2 méthodes à compléter.

```java

    //renvoie l'ensemble des personnes qui se sont inscrites a 2 conferences
    public EnsemblePersonnes inscritesA2Conferences(){
        //TODO
        EnsemblePersonnes deuxConf = new EnsemblePersonnes() ;
        for (Personne p:this.ensemblePersonnes) {
            int nbConferences= 0 ;
            for (Conference co:this.ensembleConferences) {
                if (this.contient(p,co)) {
                    nbConferences++ ;
                    if (nbConferences>2) {
                        break ;
                    }
                }
            }
            if (nbConferences==2) {
                deuxConf.ajouter(p) ;
            }
        }
        return deuxConf ;

    }

    //verifie si les 2 personnes ont au moins une conference en commun
    public boolean vontSeCroiser(Personne p1, Personne p2){
        if(p1==null || p2==null || !ensemblePersonnes.contient(p1) || !
        ensemblePersonnes.contient(p2))
            throw new IllegalArgumentException();
        //TODO
        for (Conference co:ensembleConferences) {
            if (this.ensembleCouplesPC.contient(new CouplePC(p1,co))&&this.
            ensembleCouplesPC.contient(new CouplePC(p2,co))) {
                return true ;
            }
        }

        return false;
    }

    public String toString(){
        return ensemblePersonnes+ "\n"+ ensembleConferences + "\n"+ ensembleCouplesPC;
    }

}

```

Nom : Prénom :

Question 5 (15 pts)

On vous demande d'implémenter la méthode

```
int nbLignesNonNulles()
```

de la classe `Matrice`.

Cette méthode va renvoyer le nombre de lignes non nulles de la matrice courante c-à-d le nombre de ligne ayant au moins un élément différent de 0.

Exemples :

1) Si this = $\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}$ alors la méthode `nbLignesNonNulles()` va renvoyer 1 car cette matrice n'a qu'une ligne non nulle : la deuxième.

2) Si this = $\begin{pmatrix} 0 & 0 & 3 & 0 & -3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 4 & 0 & 4 \end{pmatrix}$ alors la méthode `nbLignesNonNulles()` va renvoyer 3 car cette matrice a 3 lignes non nulles : la première, la troisième et la quatrième.

3) Si this = $\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ alors la méthode `nbLignesNonNulles()` va renvoyer 2 car cette matrice a 2 lignes non nulles : la première et la troisième.

4) Si this = $\begin{pmatrix} 0 & 1 & 0 & -1 \\ 0.5 & 0 & 1 & 2 \\ 0 & 0 & 4 & 0 \\ -2 & 0 & 0 & 2 \\ -2 & 0 & 0 & 4 \end{pmatrix}$ alors la méthode `nbLignesNonNulles()` va renvoyer 5 car cette matrice a toutes ses lignes qui sont non nulles.

5) Si this = $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ alors la méthode `nbLignesNonNulles()` va renvoyer 0 car cette matrice n'a aucune ligne non nulle.

6) Si this = $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ alors la méthode `nbLignesNonNulles()` va renvoyer 0 car cette matrice n'a aucune ligne non nulle.

7) Si this = $\begin{pmatrix} 0 & 0 & -0.5 & 0 & 0 \end{pmatrix}$ alors la méthode `nbLignesNonNulles()` va renvoyer 1 car cette matrice a son unique ligne non nulle.

8) Si this = $\begin{pmatrix} 0 \\ -0.2 \\ 1 \\ 0 \end{pmatrix}$ alors la méthode `nbLignesNonNulles()` va renvoyer 2 car cette matrice a 2 lignes non nulles : la deuxième et la troisième.

```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 ;
    }

    // Renvoie le nombre de lignes non nulles (contenant au moins un élément
    différent de 0)
    // Exemple : voir énoncé
    public int nbLignesNonNulles() {
        // TODO
        int nbLiNonNulles = 0 ;
        for (int i=0 ; i<nbLignes ; i++)  {
            int  j=0 ;
            while (j<nbColonnes && data[i][j]==0) {
                j++ ;
            }
            if (j<nbColonnes) {
                nbLiNonNulles++ ;
            }
        }
        return nbLiNonNulles ;
    }

    // affiche la matrice en format standard //NE PAS MODIFIER CETTE METHODE !!!
    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++)
                    st = st + " " ;
            }
            if (i<nbLignes-1)
                st = st+'\n' ;
        }
        return st ;
    }

}
```