TD7 - Structures et tableaux

Inès de Courchelle - Peio Loubiere

2024-2025

Objectifs :

- Manipuler des structures dans un algorithme
- Définir ses propres structures de données
- Stocker des données dans un tableau
- Utiliser des matrices dans un algorithme
- Comprendre la différence entre des fonctions, des procédures, des structures et un programme principal

Consignes :

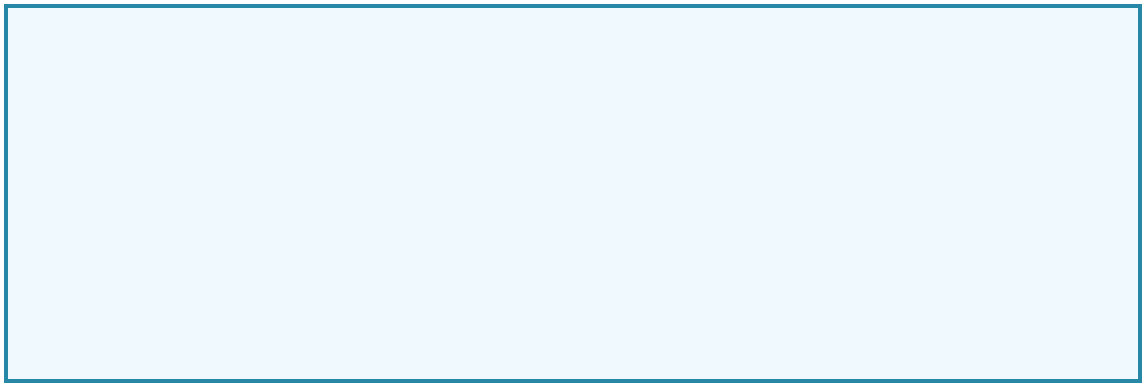
- L'ensemble des exercices ci-dessous ne seront pas tous corrigés en cours !
- Les éléments de correction seront donnés en TD, EN AUCUN CAS, des corrections toutes faites vous seront données ou distribuées. Vous devez prendre des notes !
- Durant ce TD, l'utilisation d'un papier et d'un crayon sont fortement conseillés!

Conseils
Vous pouvez utiliser d'autres feuilles de notes pour prendre la solution si vous n'avez pas assez de place !
INTERDIT DE PRENDRE EN PHOTO LES CORRECTIONS AU TABLEAU
Durée 3h00
Format papier

Exo 1 : Nombres rationels

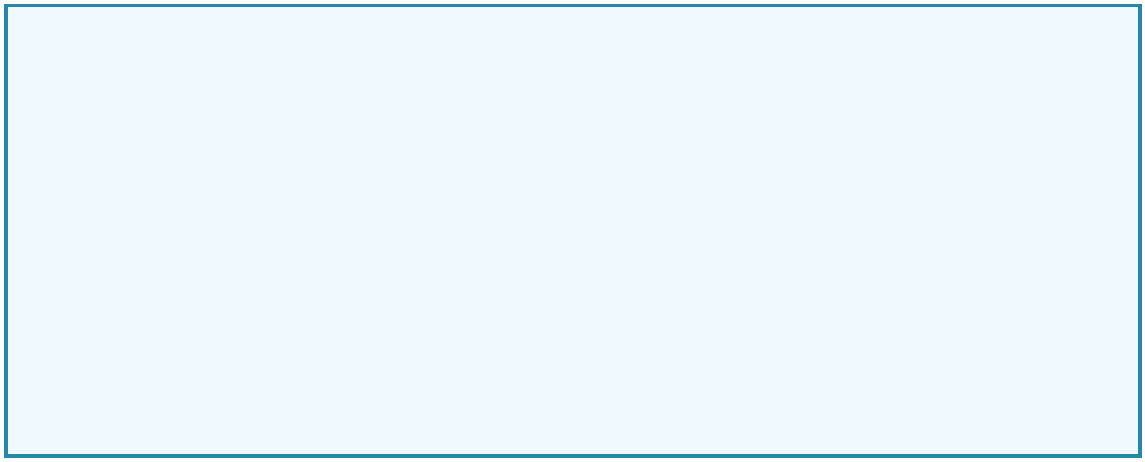
1. Définir un type rationnel permettant de représenter les nombres rationnels sous formes irréductible
2. Définir une fonction creerRatio qui prend en entrée deux entiers et retourne un rationnel. **Attention** à bien créer un rationnel sous sa forme irréductible et à bien térer les entiers négatifs … On considérera la procédure erreur("message") qui termine le programme en affichant le message d'erreur spécifié.
3. Écrire une fonction fois qui réalise le produit de deux rationnnels
4. Écrire une fonction plus qui réalise la somme de deux rationnnels
5. Écrire un prédicat qui vérifie qu'un rationnel représente un entier
6. Écrire une procédure qui affiche un rationnel.

1. Définir un type rationnel permettant de représenter les nombres rationnels sous formes irréductible

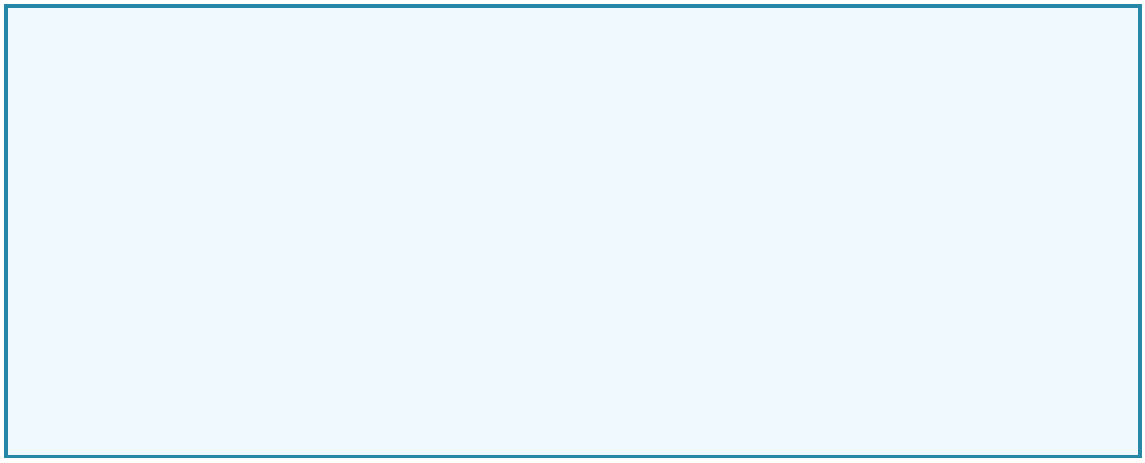


2. Définir une fonction creerRatio qui prend en entrée deux entiers et retourne un rationnel. **Attention** à bien créer un rationnel sous sa forme irréductible et à bien térer les entiers négatifs … On considérera la procédure erreur(“message”) qui termine le programme en affichant le message d’erreur spécifié.

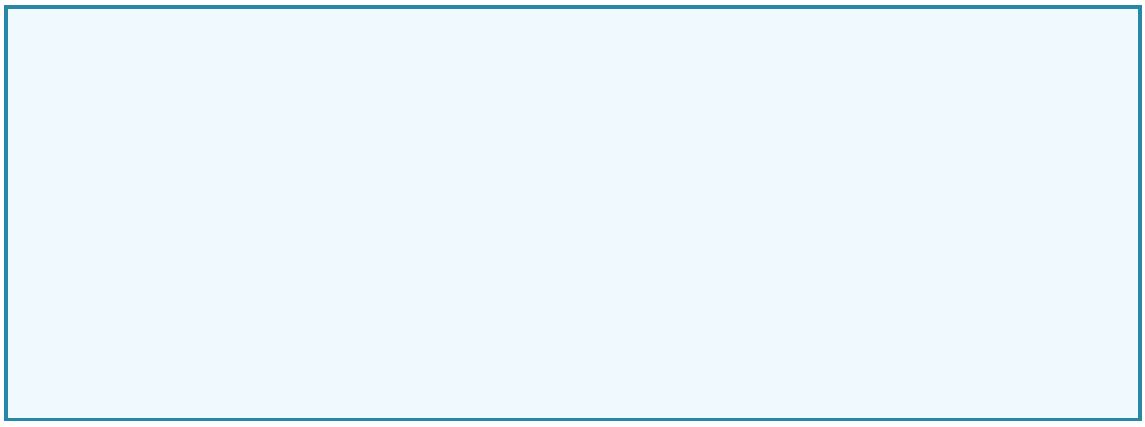
3. Écrire une fonction fois qui réalise le produit de deux rationnnels



4. Écrire une fonction plus qui réalise la somme de deux rationnnels



5. Écrire un prédicat qui vérifie qu'un rationnel représente un entier



6. Écrire une procédure qui affiche un rationnel.

Exo 2 : Tableaux et matrices statiques

1. Ecrire une procédure afficherTab qui permet d'afficher un tableau d'entier de taille N comme suit :

```
[ 8 | 42 | 24 | 12 | 17 ]
```

2. Ecrire une procédure afficherMatrice qui permet d'afficher une matrice d'entier de taille NxMcomme suit :

```
[ 8 | 42 | 24 ]
[ 4 | 25 | 12 ]
[ 7 |  8 | 14 ]
```

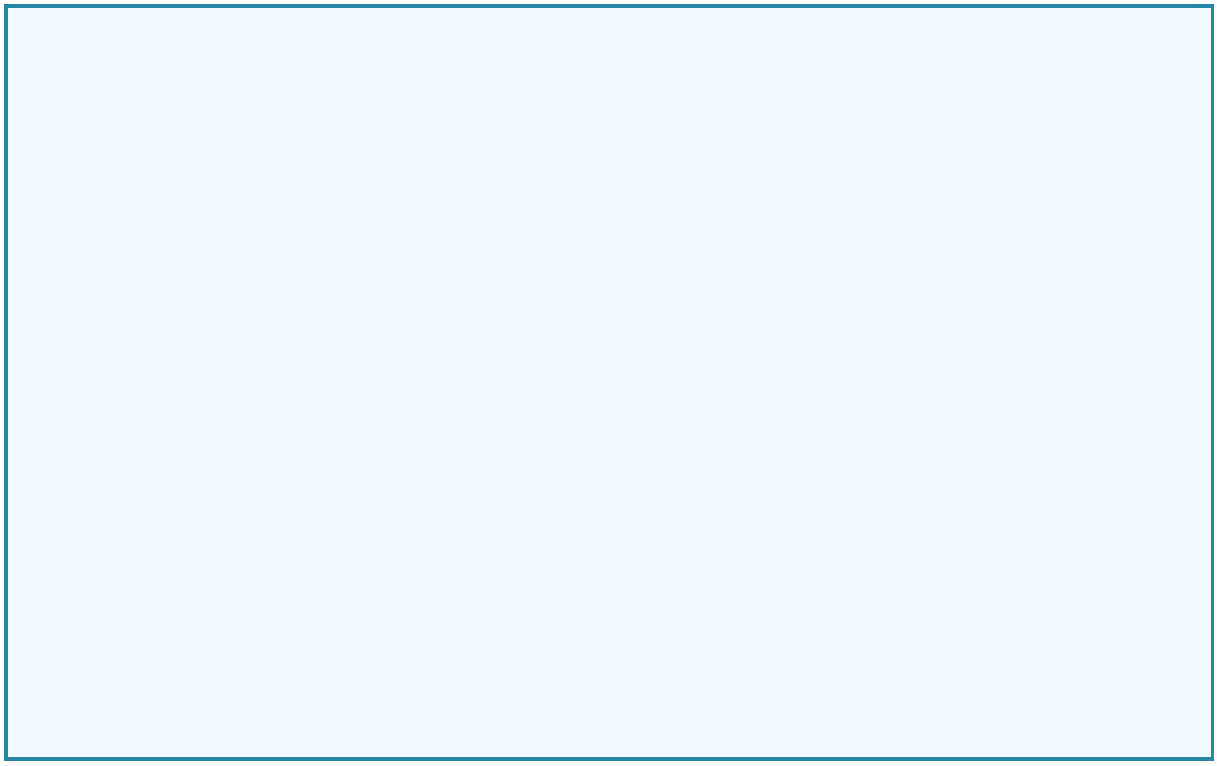
3. Écrire un prédicat qui permet de vérifier qu'un tableau 1D de taille N ne contient que des entiers de même signe.

1. Ecrire une procédure afficherTab qui permet d'afficher un tableau d'entier de taille N comme suit :

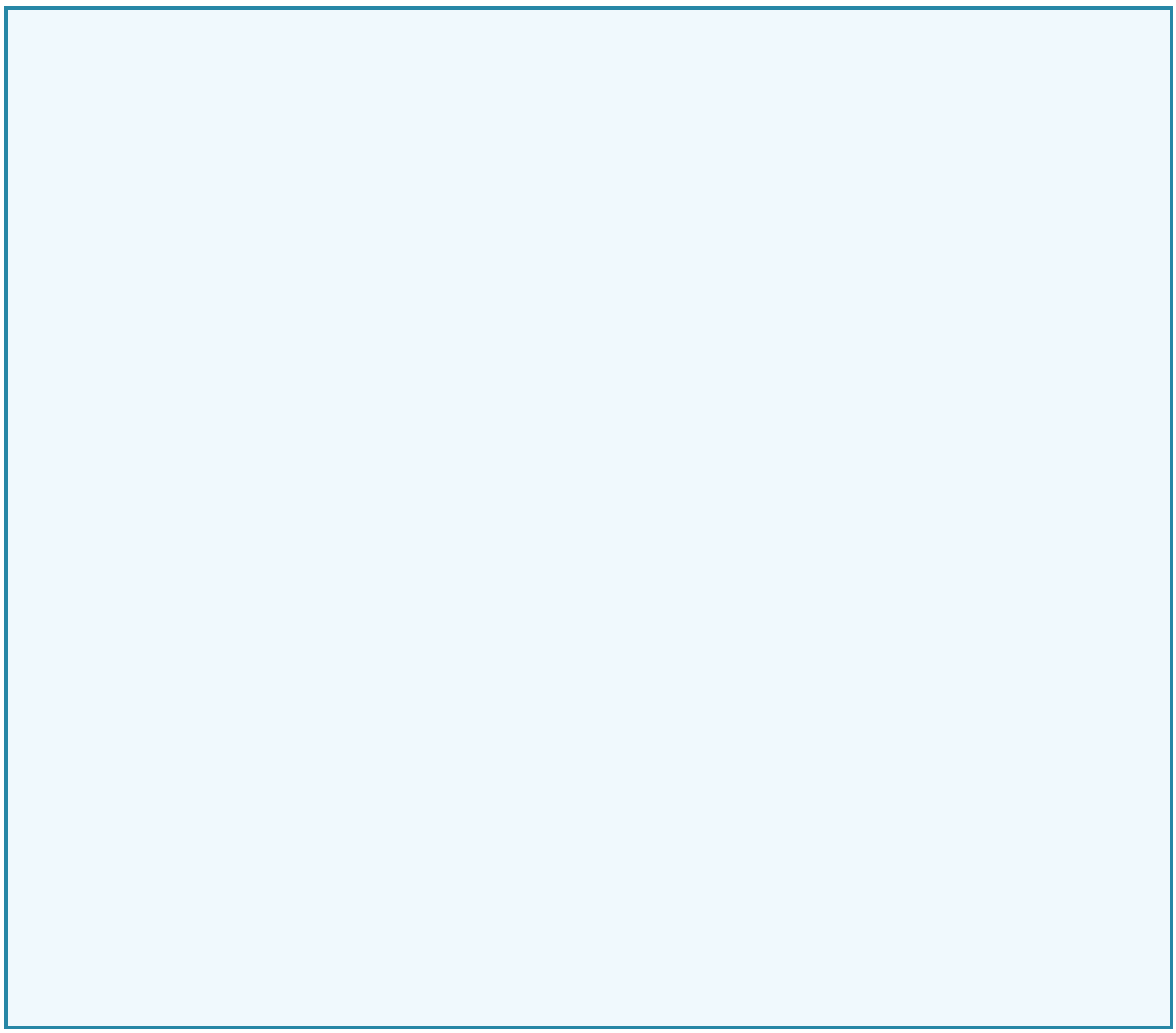
```
[ 8 | 42 | 24 | 12 | 17 ]
```

2. Ecrire une procédure afficherMatrice qui permet d'afficher une matrice d'entier de taille NxM- comme suit :

```
[ 8 | 42 | 24 ]
[ 4 | 25 | 12 ]
[ 7 |  8 | 14 ]
```



3. Écrire un prédicat qui permet de vérifier qu'un tableau 1D de taille N ne contient que des entiers de même signe.



Exo 3 : Schmiblick

Pour calculer le *schmilblick*, il faut multiplier chaque élément du tableau 1 par chaque élément du tableau 2, et additionner le tout.

Par exemple :

```
    t1 : {4, 8, 7, 12}
    t2 : {3, 6}
    schmilblick(t1, t2) = 3 * 4 + 3 * 8 + 3 * 7 + 3 * 12 + 6 * 4 + 6 * 8 +
        6 * 7 + 6 * 12 = 279
```

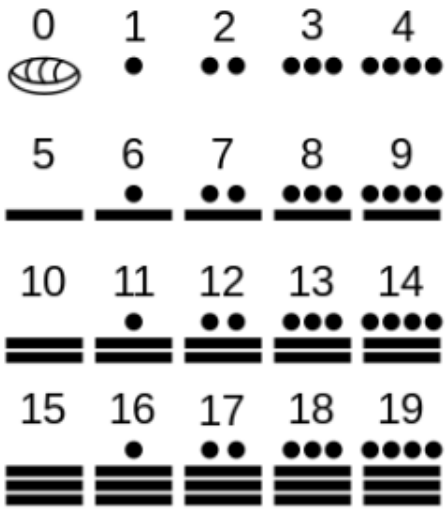
À partir de deux tableaux de tailles N et M, écrire une fonction schmilblick qui calcule le *schmilblick* des deux tableaux.

Exo 4 : Dans l'autre sens

Écrire une procédure retourne qui inverse (au sens de l'ordre) les valeurs d'un tableau de taille N. Par exemple : retourne({4, 8, 7, 12}) = {12, 7, 8, 4}

Exo 5 : Nombres Mayas

La numération maya est une numération de position de base 20 (à une irrégularité près dans la notation des grandes durées).” Le système est simple (cf figure 1), basé sur deux symboles : le point désigne une unité, le trait pour 5 points



1. Créer un type structuré “maya”, composé d’un champ trait et d’un champ point.
2. Créer un constructeur (*une fonction*) creerMaya pour ce type (on considérera seulement les nombres de 0 à 19 !). Si les paramètres d’entrée sont invalides, le constructeur devra déclencher une erreur.
3. Réaliser l’opération plus entre nombres maya (sans passer par notre numérotation…). **Attention :** si le résultat du calcul atteint les 20aines ou 30aines, le résultat sera une valeur non conforme et une erreur devra se déclencher.

1. Créer un type structuré “maya”, composé d’un champ trait et d’un champ point.

2. Créer un constructeur (*une fonction*) creerMaya pour ce type (on considérera seulement les nombres de 0 à 19 !). Si les paramètres d’entrée sont invalides, le constructeur devra déclencher une erreur.

3. Réaliser l'opération plus entre nombres maya (sans passer par notre numérotation…). **Attention :** si le résultat du calcul atteint les 20aines ou 30aines, le résultat sera une valeur non conforme et une erreur devra se déclencher.

Exo 6 : Le carré magique

Un carré magique d'ordre n est un tableau n x n tel que la somme des entiers de chaque ligne, chaque colonne et des deux diagonales est identique.

Exemple : Tableau carré magique d'ordre 3

8	1	6
3	5	7
4	9	2

Écrire un prédicat estMagique qui vérifie qu'un carré d'entiers (d'ordre n) est magique. Pour cela, nous allons réaliser plusieurs fonctions pour vérifier si un carré est magique :

1. Écrire une fonction permettant de faire la somme d'une ligne passée en paramètre
2. Écrire une fonction permettant de faire la somme d'une colonne passée en paramètre
3. Écrire une fonction permettant de faire la somme de la diagonale du carré en partant d'en bas à gauche
4. Écrire une fonction permettant de faire la somme de la diagonale du carré en partant d'en haut à gauche
5. Écrire un prédicat permettant de dire si un carré passé en paramètre est magique. Vous utiliserez les fonctions précédement créées

```
// Fonction permettant de faire la somme d'une ligne passée en paramé
   tre
// Préconditions : la matrice, sa taille et la ligne à compter
// Postconditions : la somme de la ligne
FONCTION cptLigne(carre : matrice d'entiers, N,ligne : entier) : entier
```

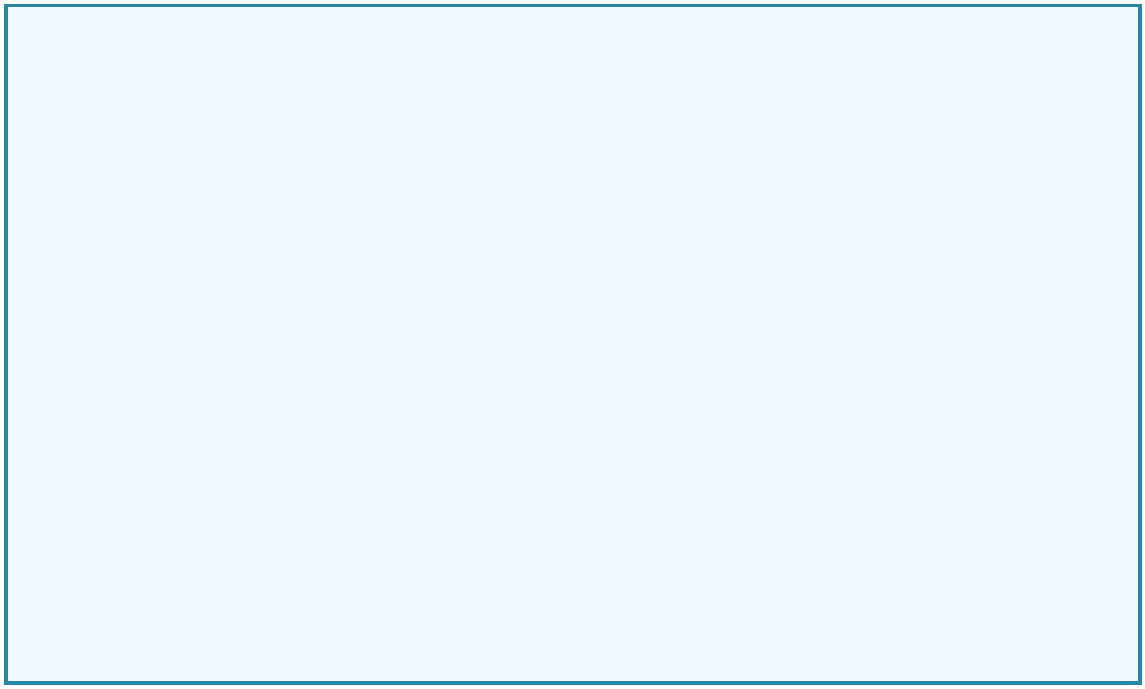
```
// Fonction permettant de faire la somme d'une colonne passée en paramé
   tre
// Préconditions : la matrice, sa taille et la colonne à compter
// Postconditions : la somme de la colonne
FONCTION cptColonne(carre : matrice d'entiers, N,colonne : entier) :
   entier
```

```
// Fonction permettant de faire la somme de la diagonale du carré en
   partant d'en bas à gauche
// Préconditions : la matrice, et sa taille
// Postconditions : la somme de la diagonale
FONCTION cptDiag1(carre : matrice d'entiers, N : entier) : entier
```

```
// Fonction permettant de faire la somme de la diagonale du carré en
   partant d'en haut à gauche
// Préconditions : la matrice, et sa taille
// Postconditions : la somme de la diagonale
FONCTION cptDiag2(carre : matrice d'entiers, N : entier) : entier
```

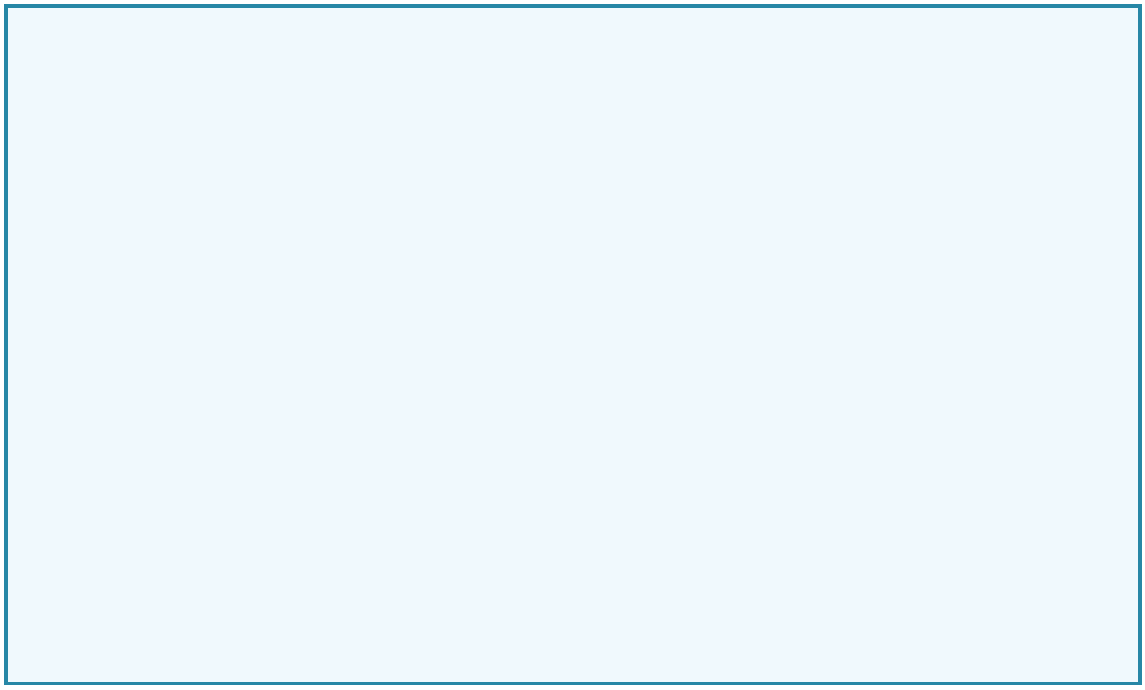
```
// Prédicat permettant de dire si un carré passé en paramétre est
   magique
// Préconditions : la matrice, et sa taille
// Postconditions : vrai => si le carré est magique, faux => si le carr
   é n'est pas magique
FONCTION estMagique(carre : matrice d'entiers, N : entier) : booleen
```

1. Écrire une fonction permettant de faire la somme d'une ligne passée en paramètre

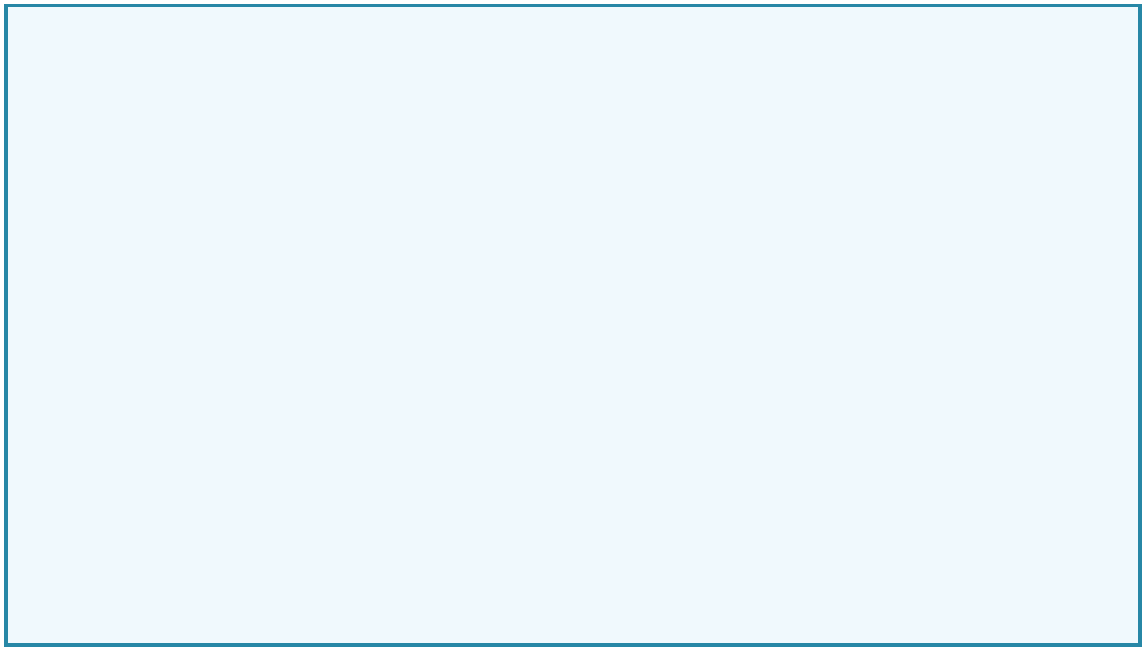


2. Écrire une fonction permettant de faire la somme d'une colonne passée en paramètre

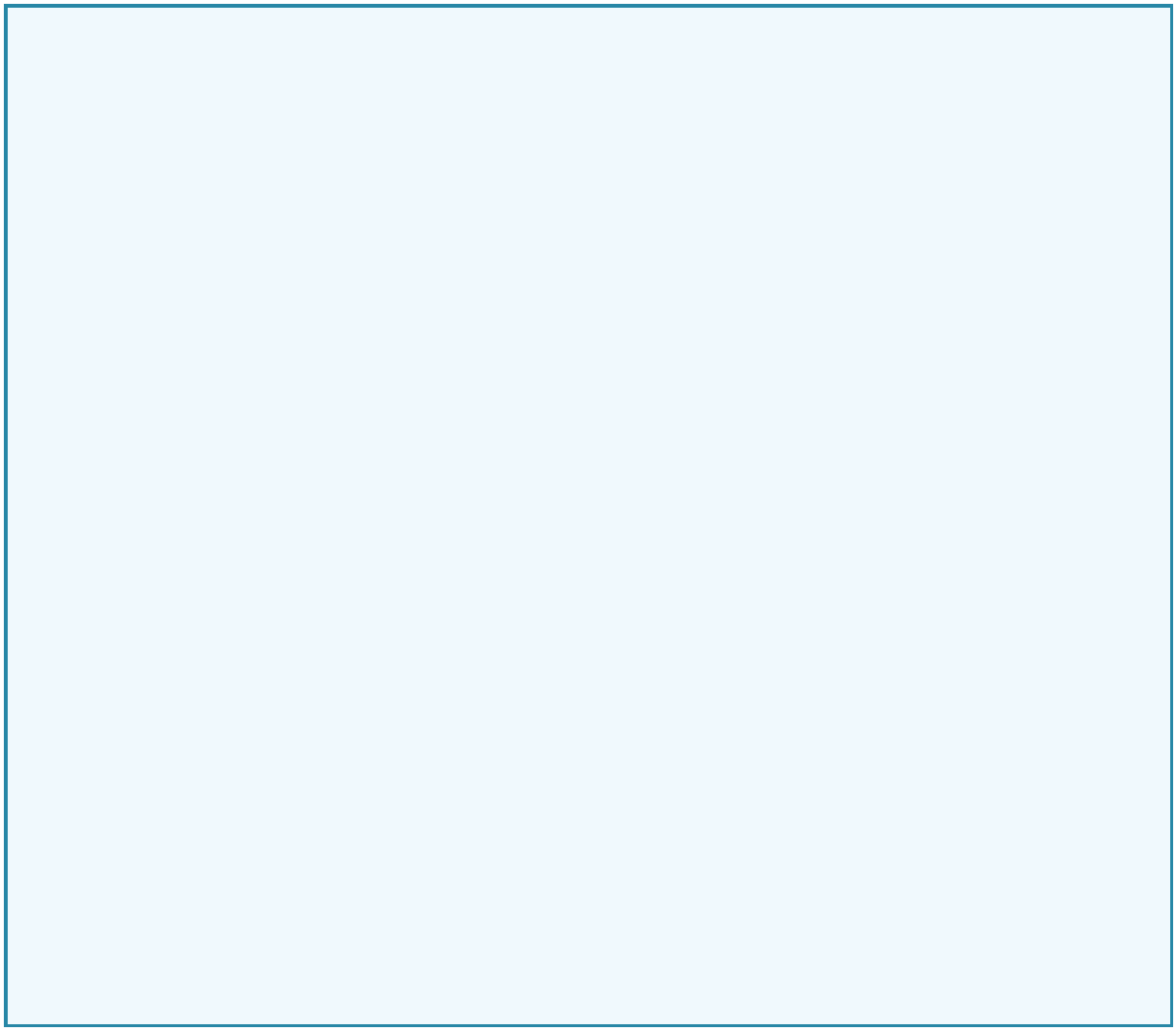
3. Écrire une fonction permettant de faire la somme de la diagonale du carré en partant d'en bas à gauche



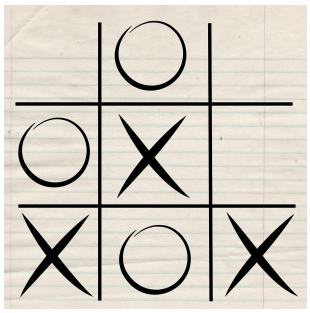
4. Écrire une fonction permettant de faire la somme de la diagonale du carré en partant d'en haut à gauche



5. Écrire un prédicat permettant de dire si un carré passé en paramètre est magique. Vous utiliserez les fonctions précédement créées.



Exo 7 : Le morpion



Le morpion se joue dans une grille à deux dimensions 3x3. Une case de la grille peut contenir 3 valeurs possibles : vide, croix ou rond.

Le jeu se joue à deux joueurs, le plus âgé choisit les croix ou les ronds. S'ils ont le même âge, c'est l'ordre alphabétique de leur prénom qui prévaut. S'ils sont identiques, leur symbole est tiré aléatoirement.

Chaque joueur joue alternativement sur une case vide. Le premier qui a 3 symboles alignés horizontalement, verticalement ou en diagonale a gagné. Nous allons définir les algorithmes permettant de coder ce jeu dans sa version deux joueurs.

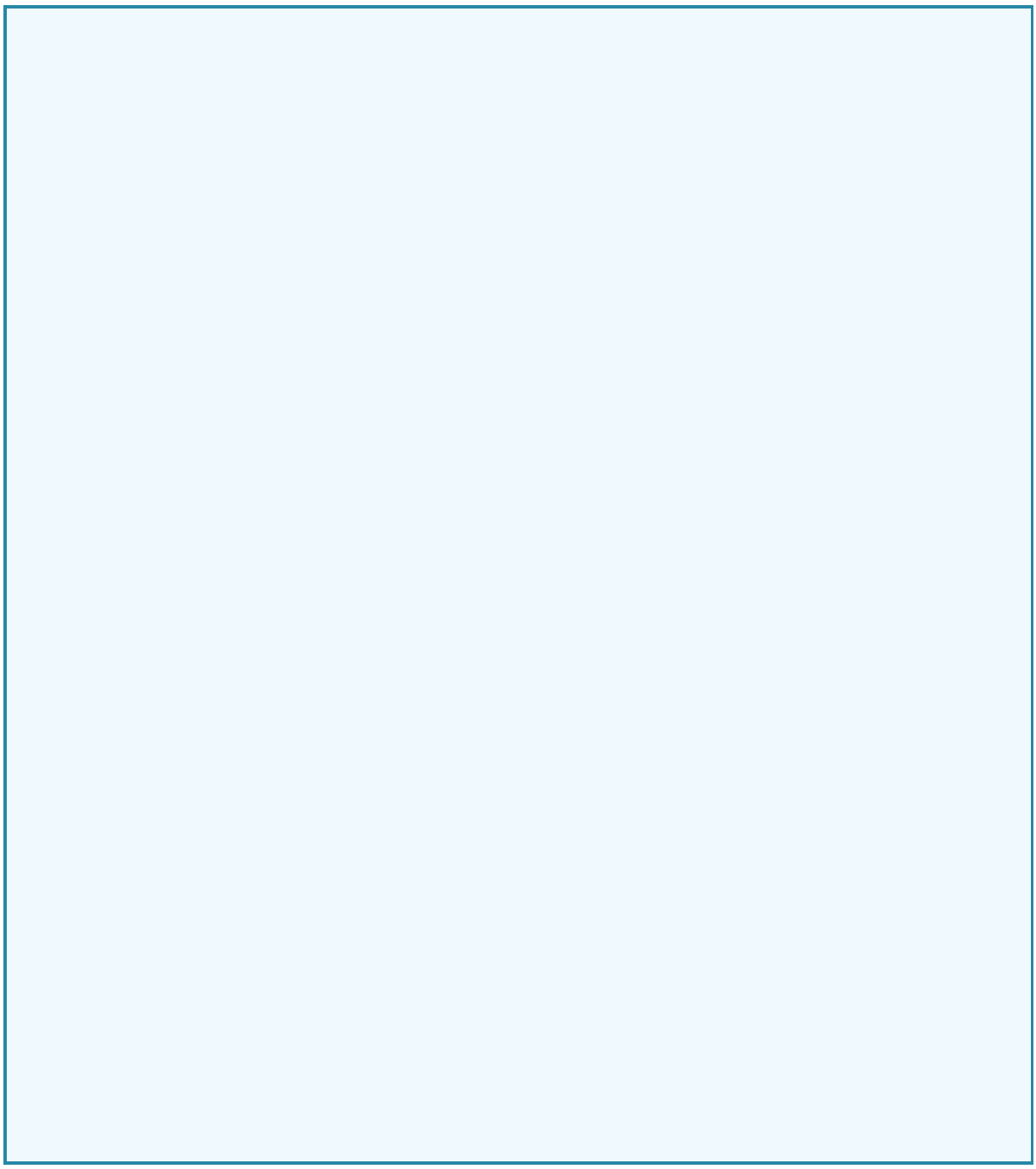
1. Créer une procédure affGrille qui permet d'afficher la grille, selon les valeurs contenues dans chaque case :

- Si la case contient un -1, alors j'affiche un espace
- Si la case contient un 1, alors j'affiche un 'X'
- Si la case contient un 2, alors j'affiche un 'O'

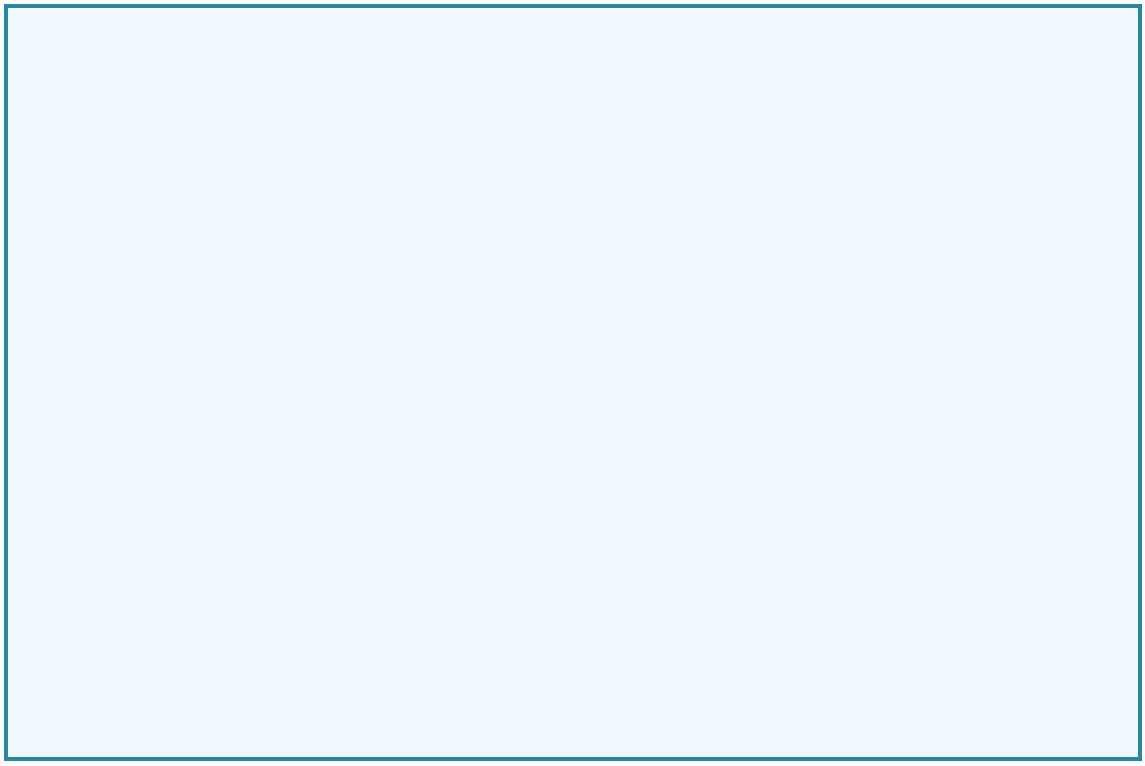
2. Créer une procédure initGrille qui permet d'initialiser la grille, lorsqu'aucun joueur n'a joué. Elle contiendra uniquement la valeur -1
3. Créer une procédure tour2jeu qui prend en paramètre la grille de jeu, le joueur à jouer, qui demande au joueur de choisir une case vide (ligne+colonne, à vérifier), et si la case est bien vide qui joue le coup.
4. Écrire une fonction gagne qui prend en paramètre un joueur, la grille et qui vérifie si le joueur a gagné
5. Créer le programme principal qui affiche la grille puis alterne les tours de jeu jusqu'à ce qu'un joueur gagne ou que toutes les cases soient remplies, sans vainqueur. Vous afficherez alors le nom du vainqueur ou bien "match nul"

1. Créer une procédure affGrille qui permet d'afficher la grille, selon les valeurs contenues dans chaque case :

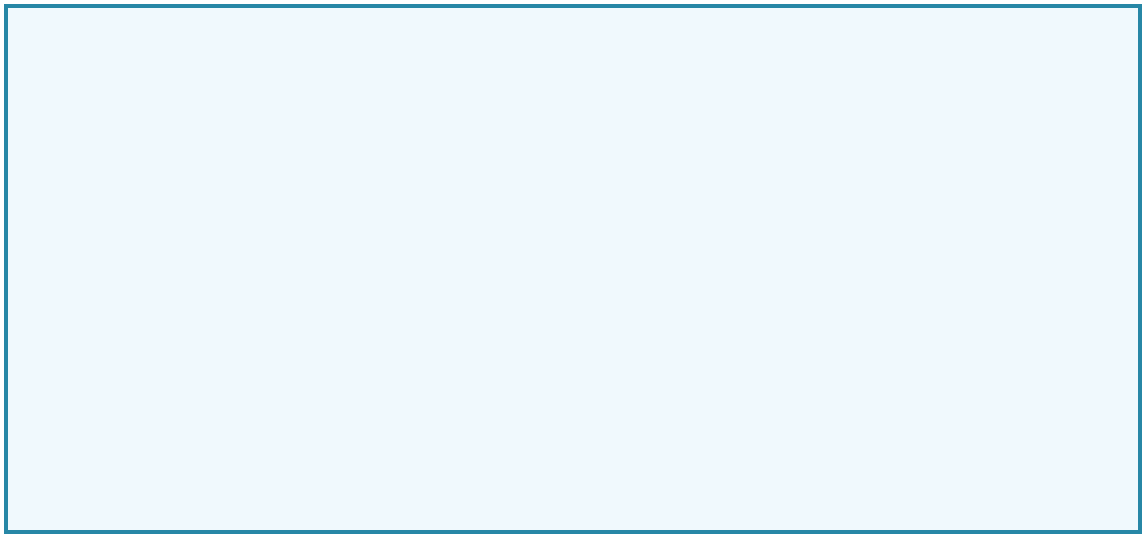
- Si la case contient un -1, alors j'affiche un espace
- Si la case contient un 1, alors j'affiche un 'X'
- Si la case contient un 2, alors j'affiche un 'O'



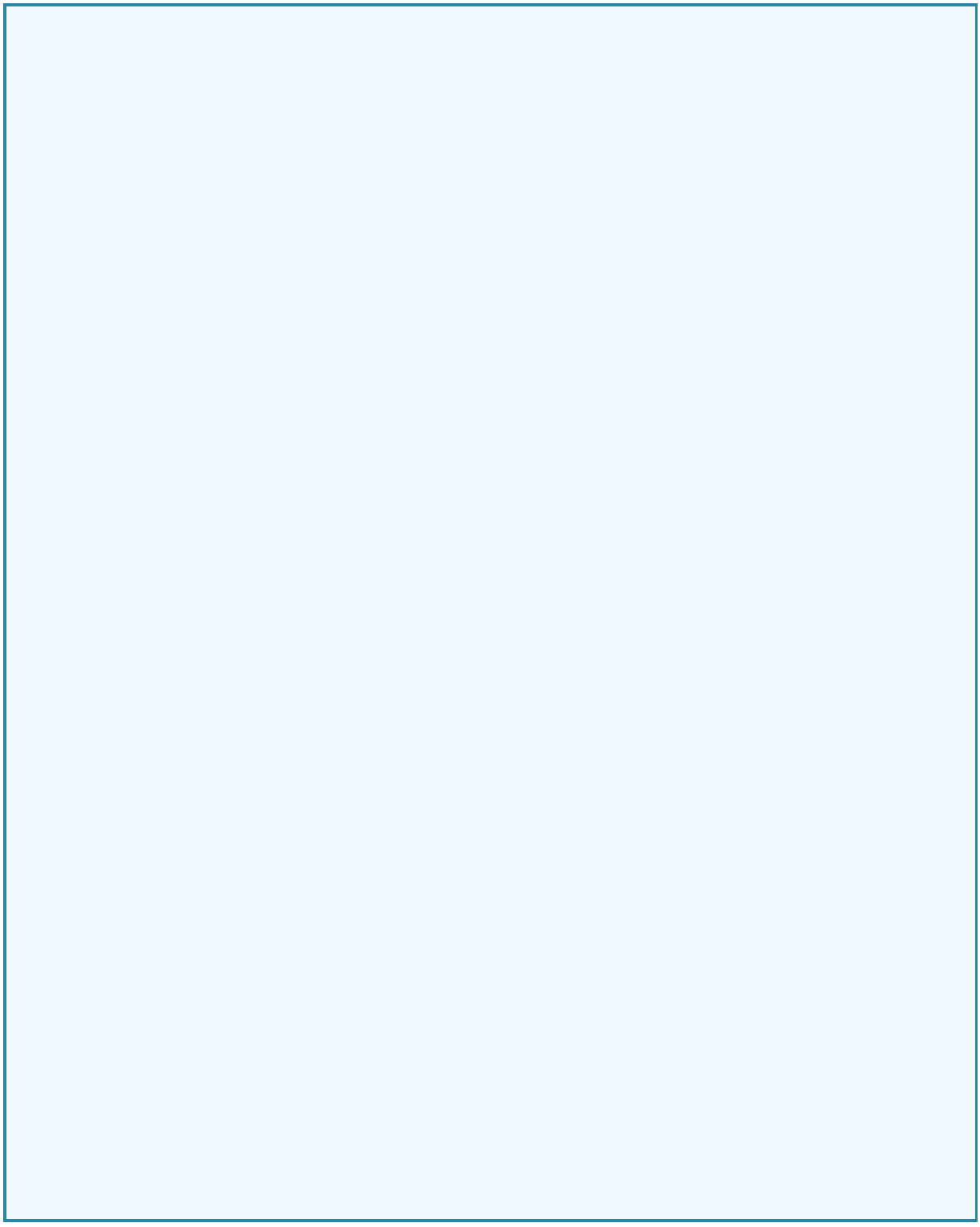
2. Créer une procédure initGrille qui permet d'initialiser la grille, lorsqu'aucun joueur n'a joué. Elle contiendra uniquement la valeur -1



3. Créer une procédure tour2jeu qui prend en paramètre la grille de jeu, le joueur à jouer, qui demande au joueur de choisir une case vide (ligne+colonne, à vérifier), et si la case est bien vide qui joue le coup.



4. Écrire une fonction gagne qui prend en paramètre un joueur, la grille et qui vérifie si le joueur a gagné



5. Créer le programme principal qui affiche la grille puis alterne les tours de jeu jusqu'à ce qu'un joueur gagne ou que toutes les cases soient remplies, sans vainqueur. Vous afficherez alors le nom du vainqueur ou bien "match nul"