Carrément magique !

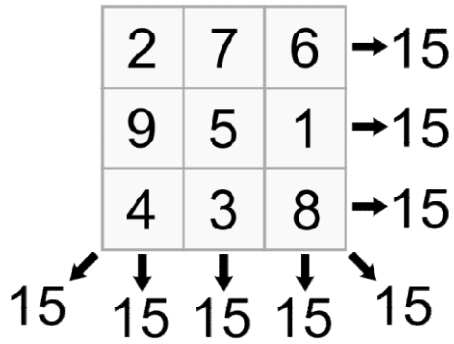
Consignes : vous avez le droit d'utiliser vos ordinateurs personnels ou ceux de l'ENSSAT. Vous pouvez utiliser tous les documents de cours et internet. L'usage d'IA est interdit.

Bon courage !

Qu'est ce qu'un carré magique

On appelle carré magique d'ordre n un tableau multidimensionnel carré contenant n^2 entiers positifs. La propriété fondamentale d'un carré magique est que ***la somme*** des éléments composants une ***ligne*** quelconque, ou celle des composants une ***colonne*** quelconque ou d'une ***diagonale*** quelconque donne un nombre unique appelé **constante magique**.

Exemple :



(Source Wikipédia).

Passons maintenant au travail de codage. Il est demandé d'utiliser le plus exhaustivement possible les syntaxes amenées par la version es6 de JavaScript.

Vous aurez les fichiers suivants :

- index.html qui contient votre code html de base,
- styles.css pour présenter votre interface graphique,
- main.js qui contient l'instance de Vue gérant l'affichage dynamique de votre interface,
- matrix.js qui contient la classe Matrix qui contient des fonctions utiles pour la manipulation des matrices,
- magic_square.js qui contient la classe MagicSquare qui permet de manipuler les carrés,
- magiques selon les indications données plus loin.

Il est demandé de faire tous les liens nécessaires dans le fichier index.html via les balises <script …> </script> et autres. Vue.js version 3 est utilisé via le cdn donné en cours.

Vous pourrez bien sûr rajouter des fichiers selon le besoin –le tout est que la structure soit claire !

Nous commencerons par créer une classe de calcul matriciel qui pourra être utilisée pour effectuer certains calculs dans la classe MagicSquare. Ces classes sont décrites par la suite.

I- Ecriture de la classe Matrix

Cette classe est assez générique pour pouvoir être utilisées pour d'autres problèmes sur les matrices. Les matrices seront représentées en JavaScript par des tableaux à deux dimensions.

Nous partirons sur le code suivant :

```
class Matrix {
 constructor(data) {
  this.data = data;
  this.rows = data.length;
  this.cols = data[0].length;
 }
  //... specific matrix methods
}
```

Avec cette définition, vous devriez pouvoir tester vos fonctions de la façon suivante :

```
const matrix = new Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]]);
console.log(matrix.isSquare()); // true
console.log(matrix.trace()); // 15
console.log(matrix.transpose().display()); // Affiche la
matrice transposée
etc.
```

1- Ecrire une fonction ***isSquare*** qui vérifie si une matrice est carrée. Une matrice est dite carrée si elle a le même nombre de lignes que de colonnes. Cette fonction renvoie true ou false.

2- Ecrire une fonction ***display*** qui permet d'afficher une matrice carrée sous la forme d'un tableau du nombre de lignes et de colonnes voulues dans la console. Elle sera utilisée avec profit pour les vérifications.

3- Ecrire une fonction ***add ()*** qui permet de sommer deux matrices carrées. Cette fonction admet comme paramètre une autre matrice de caractéristiques convenables. Pour effectuer une addition de deux matrices, il suffit d'additionner les différents éléments de même position dans les deux matrices. On obtient ainsi une matrice somme de même dimension que les matrices carrées de départ qui sera renvoyée par la fonction.

Exemple : $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} + \begin{bmatrix}4 & 5\\6 & 7\end{bmatrix} = \begin{bmatrix}5 & 7\\9 & 11\end{bmatrix}$

4- Ecrire une fonction ***subtract ()*** qui permet de soustraire deux matrices carrées. Il suffit dans ce cas de soustraire les différents éléments de même position dans les deux matrices. On obtient une matrice de même dimension que les matrices carrées de départ.

Exemple : $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} - \begin{bmatrix}4 & 5\\6 & 7\end{bmatrix} = \begin{bmatrix}-3 & -3\\-3 & -3\end{bmatrix}$

5- Ecrire une fonction ***trace*** qui permet de trouver la trace d'une matrice carrée, c'est dire la somme des nombres qui se trouvent sur la diagonale principale.

Exemple : la trace de la matrice $\begin{bmatrix}\mathbf{1} & 2 & 3\\4 & \mathbf{5} & 6\\7 & 8 & \mathbf{9}\end{bmatrix}$ est 1+5+9 =15.

6- Ecrire une fonction ***antiTrace*** (désolé pour les puristes) qui effectue la somme des des nombres qui se trouvent sur la diagonale secondaire.

Exemple : la trace de la matrice $\begin{bmatrix}1 & 2 & \mathbf{3}\\4 & \mathbf{5} & 6\\\mathbf{7} & 8 & 9\end{bmatrix}$ est 3+5+7 =15.

7- Ecrire une fonction ***transpose*** qui transpose d'une matrice carrée, c'est-à-dire la matrice qui est symétrique par rapport à sa diagonale principale. Cette matrice transposée est renvoyée par la fonction.

Exemple : la matrice transposée de $\begin{bmatrix}\mathbf{1} & 2 & 3\\4 & \mathbf{5} & 6\\7 & 8 & \mathbf{9}\end{bmatrix}$ est $\begin{bmatrix}1 & 4 & 7\\2 & 5 & 8\\3 & 6 & 9\end{bmatrix}$

8- Ecrire la fonction ***isIdentical ()*** qui vérifie si deux matrices carrées sont identiques, c'est-à-dire si elles possèdent exactement les mêmes coefficients sur les différentes lignes et colonnes. La fonction renvoie un booléen selon le résultat.

Exemple : A = $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$ et $B = \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$ sont identiques.

9- Ecrire une fonction ***dilate ()*** qui permet de trouver la matrice dilatée d'une matrice carrée X. Pour trouver la matrice dilatée d'une autre matrice il suffit de multiplier chacun des coefficients de la matrice X par un nombre k.

Exemple : la matrice dilatée de $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$ par k=2 est $\begin{bmatrix}2 & 4 & 6\\8 & 10 & 12\\14 & 16 & 18\end{bmatrix}$

10- Ecrire la fonction ***isSymetric*** qui renvoie true si la matrice est symétrique et false sinon.
Une matrice est dite symétrique si les éléments situés symétriquement par rapport à la diagonale principale sont identiques.

Exemple : la matrice $\begin{bmatrix} \mathbf{1} & 2 & 3 \\ 2 & \mathbf{5} & 6 \\ 3 & 6 & \mathbf{9} \end{bmatrix}$ est symétrique.

11- Ecrire la fonction ***isAntisymetric*** qui renvoie true si la matrice est antisymétrique et false sinon. Une matrice est dite antisymétrique qui les éléments situés symétriquement par rapport à la diagonale principale sont opposés.

Exemple : la matrice $\begin{bmatrix} \mathbf{0} & -2 & -3 \\ 2 & \mathbf{0} & 6 \\ 3 & -6 & \mathbf{0} \end{bmatrix}$ est antisymétrique.

12- Ecrire une fonction ***multiply ()*** qui permet de multiplier deux matrices carrées

Exemple : $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 2 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 21 & 33 & 45 \\ 51 & 81 & 111 \\ 81 & 129 & 177 \end{bmatrix}$

13- On dit qu'une matrice est triangulaire supérieure si tous les éléments strictement en dessous de la diagonale principale sont nuls.

Exemple : $\begin{bmatrix} 1 & 2 & 3 \\ \mathbf{0} & 5 & 6 \\ \mathbf{0} & \mathbf{0} & 9 \end{bmatrix}$ est une matrice triangulaire supérieure.

A- Ecrire une fonction **isUpperTriangular** qui renvoie true si X est triangulaire supérieure et false sinon.

B- Ecrire une fonction ***upperTriangularSum*** qui retourne la somme de tous les éléments au dessus de la diagonale d'une matrice quelconque.

Exemple : La somme triangulaire supérieure de la matrice $\begin{bmatrix} 1 & \mathbf{2} & \mathbf{3} \\ 4 & 5 & \mathbf{6} \\ 7 & 8 & 9 \end{bmatrix}$ est égale à 11.

14- On dit qu'une matrice est triangulaire inférieure si tous les éléments strictement au dessus de la diagonale principale sont nuls.

Exemple : $\begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} \\ 1 & 5 & \mathbf{0} \\ 2 & 3 & 9 \end{bmatrix}$ est une matrice triangulaire supérieure.

A- Ecrire une fonction **isLowerTriangular** qui renvoie true si X est triangulaire inférieure et false sinon.

B- Ecrire une fonction ***lowerTriangularSum*** qui retourne la somme de tous les éléments au dessous de la diagonale d'une matrice quelconque.

Exemple : La somme triangulaire supérieure de la matrice $\begin{bmatrix} 1 & \mathbf{2} & \mathbf{3} \\ 4 & 5 & \mathbf{6} \\ 7 & 8 & 9 \end{bmatrix}$ est égale à 11.

II- Ecriture de la classe MagicSquare

```
class MagicSquare {
    constructor(n) {
        this.taille = n;
        // initialise the matrix here
    }
}
```

1- Compléter le constructeur de la classe ***MagicSquare*** pour que lors de la création d'un carré magique celui-ci ait le nombre -1 dans chacune de ses cases.

2- Ecrire une fonction ***isMagic*** qui renvoie la constante magique du carré si celui-ci est magique et – 1 sinon.

3- Ecrire une fonction ***contains ()*** qui renvoie true si le nombre nb est présent dans le tableau et false sinon

Exemple : dans le tableau précédent au début de l'énoncé, ***contains*** renvoie true si nb est 5 et renvoie false si nb est 13.

4- Un carré magique est dit normal d'ordre n si et seulement si tous les nombres entre 1 et n^2 sont présents dans le tableau.

Exemple : le carré magique précédent est normal car tous les nombres compris entre 1 et 9 (3^2) sont présents sans exception dans le tableau.

Ecrire une fonction ***isNormal*** qui renvoie true si le carré magique est normal et false sinon.

5- Pour additionner deux tableaux magiques, on doit additionner les cases correspondantes des 2 tableaux. Ecrire la fonction ***add ()*** qui effectue l'addition des deux carrés magiques.

6- Pour soustraire deux carrés magiques, il faut soustraire les cases correspondantes des deux carrés. Après avoir écrit la fonction ***substract ()***. Vérifier que le carré magique obtenu est magique.

7- Ecrire la fonction ***generateByLucasMethod (int a, int b, int c)*** qui génère un carré magique d'ordre 3 par la méthode de Lucas. en tenant compte des informations suivantes :

Cette fonction renvoie true si la construction du carré magique est réussie et false sinon.

Voici exposé ci-dessous la ***méthode de Lucas*** :

a- Cette méthode ne s'applique que pour des ***carrés magiques d'ordre 3***.

b- Soit a, b et c des entiers. La formule générale applicable pour remplir les carrés magiques est la suivante :

c + a	c – a – b	c + b
c – a + b	c	c + a – b
c – b	c + a + b	c – a

c- De plus, pour qu'un carré magique de dimension 3*3 puisse être généré, il faut que les entiers a, b, c vérifient les conditions suivantes : ***0<a<b<c-a*** et ***b doit être différent de 2*a***.

8- Ecrire la fonction ***generateBySiamMethod ()*** qui permet de créer des carrés magiques d'ordre impair.

Vous suivrez les indications suivantes relatives à l'a**lgorithme de Siam (ou de La Loubère) pour construire un carré magique d'ordre impair :**

a- Initialisation**:**
- Créer un tableau carré de taille n x n, où n est un nombre impair.
- Initialiser toutes les cases du tableau.
- Choisir une case de départ (par exemple, la case centrale de la première ligne).

b- Placement du premier nombre:
- Placer le nombre 1 dans la case de départ.

c- Itérations successives:
Pour chaque nombre suivant (de 2 à n^2), effectuer les opérations suivantes :
- **Déplacement:** Se déplacer d'une case vers la droite et d'une case vers le haut.
- **Gestion des bords:**
  - Si on sort du carré vers la droite, revenir à la première colonne.
  - Si on sort du carré vers le haut, aller à la dernière ligne.
- **Case occupée:**
  - Si la case où l'on veut placer le nombre est déjà occupée, descendre d'une case.
- **Placement du nombre:**
  - Placer le nombre courant dans la case déterminée.

d- **Répétition:**
- Répéter l'étape 3 jusqu'à ce que tous les nombres aient été placés.

Conseil algorithmique : modulez vos débordements !

9- Vérifier que la somme de deux carrés magiques est elle aussi magique.
10- Vérifier que la différence de deux carrés magiques est elle aussi magique.

III- Visualisation du carré magique et jeu

1- Utiliser Vue.js pour représenter simplement un carré magique facilement lisible et agréable. Vous utiliserez vous cela le fichier index.html contenant le cdn de Vue et un fichier main.js contenant une instance de Vue.
2- Effectuer les modifications nécessaires dans votre code pour que l'on puisse générer un carré magique incomplet (ne contenant pas l'intégralité des nombres) et qu'une personne puisse s'amuser à trouver les nombres manquants. On précisera à ce joueur la constante du tableau magique.