TP – Vecteurs
ING1 – Programmation C

Préambule

L'ensemble du TP formera un seul programme. Les différentes fonctionnalités traitées devront être accessible via un menu. Je vous conseille de bien séparer chacune des fonctions, ainsi que celle du menu, de la saisie ... Bien découper votre code vous facilitera la lisibilité du code, et vous évitera un certain nombre d'erreurs.

Vous travaillerez sur des tableaux de taille fixe. La taille sera donnée par une constance symbolique. Pour rappel, une constante symbolique se définit par :

```
#define M 20
```

1 Tableaux 1D – Prise en main des tableaux

(1) **Inversion**
Écrire une procédure qui inverse les éléments d'un tableau (sans passer par un tableau temporaire) : le premier devient le dernier, le deuxième devient l'avant dernier, etc. □

(2) **Somme**
Écrire une procédure qui somme les éléments de deux tableaux dans un troisième. Attention, pour le moment, vous ne savez pas retourner un tableau, donc vous devrez utiliser une procédure. □

(3) **Tableau traversable**
Un tableau est dit traversable si, en commençant à l'indice 0 et en ajoutant à chaque étape *tab[indice]* à l'indice, on arrive à la dernière case du tableau.
Exemple de tableau traversable :

3	7	2	-1	1	9

- étape 1 : indice = 0, tab[indice] = 3
- étape 2 : indice = 3, tab[indice] = -1
- étape 3 : indice = 2, tab[indice] = 2
- étape 4 : indice = 4, tab[indice] = 1
- étape 5 : indice = 5, dernière case du tableau
- retourne VRAI

Écrire la fonction qui permet de déterminer si un tableau est traversable. □

(4) **Palindrome**
Un palindrome est un mot ou une phrase dont l'ordre des lettres reste le même qu'on le lise de gauche à droite ou de droite à gauche comme dans l'expression "esope reste ici et se repose", ou le mot "radar".
Écrire une fonction qui permet de dire si un mot est un palindrome. Elle retournera 1 si c'est le cas, 0 sinon. □

2 Tableaux 2D – Puissance 4

Le jeu de Puissance 4 se joue sur une grille carrée de $N \times N$ cases (N défini préalablement). Deux joueurs s'affrontent. Ils doivent remplir chacun à leur tour une des colonnes de la grille par leur pion. Les pions seront symbolisés par des lettres : O ou X. La grille est en position verticale, ce qui fait que les pièces "tombent" au plus bas de la grille.
Par exemple, voici deux configurations :

```
    0   1   2   3   4
  +---+---+---+---+---+
0 |   |   |   |   |   |
  +---+---+---+---+---+
1 |   |   |   |   |   |
  +---+---+---+---+---+
2 |   |   | X | O |   |
  +---+---+---+---+---+
3 |   |   | O | X |   |
  +---+---+---+---+---+
4 | X | O | O | X |   |
  +---+---+---+---+---+

Position correcte des pions
```

```
    0   1   2   3   4
  +---+---+---+---+---+
0 |   |   |   |   |   |
  +---+---+---+---+---+
1 |   |   |   |   |   |
  +---+---+---+---+---+
2 |   | X | X | O |   |   <-- Erreur
  +---+---+---+---+---+
3 |   |   | O | X |   |
  +---+---+---+---+---+
4 | X | O | O | X |   |
  +---+---+---+---+---+

Position incorrecte des pions
```

Sur la grille de droite, le pion en position {2,1} est incorrecte puisqu'il devrait tomber en position {3,1}.

Les joueurs doivent remplir chacun à leur tour une colonne de la grille avec le symbole qui leur est attribué : O ou X. Le gagnant est celui qui arrive à aligner quatre symboles identiques,

horizontalement, verticalement ou en diagonale.

La structure utilisée pour modéliser le plateau sera un tableau d'entiers à deux dimensions. On utilisera un tableau statique de taille $N \times N$. Les valeurs possibles du plateau seront : -1 si personne n'a encore joué à cette case, 1 ou 2 si le joueur respectivement 1 ou 2 a joué à cette case.

Initialisation

Au tout début du jeu, les cases du plateau seront initialisées à -1, pour signifier qu'aucun joueur n'a joué. Implémenter la méthode d'initialisation :

```
void init (int ttint_plateau[N][N]);
```

Affichage

À chaque tour de jeu, le plateau sera affiché afin de visualiser l'avancement de la partie. Implémenter la méthode `affichage` qui écrit 'X' lorsque le joueur 1 a joué sur la case, 'O' s'il s'agit du joueur 2, ou alors ' ' s'il n'y a personne.

```
void affichage (int ttint_plateau[N][N]);
```

Jouer

Les joueurs jouent tour à tour. Écrire une méthode `jouer` qui permet à un joueur donné de jouer dans une colonne spécifique. Si la colonne est pleine, afficher un message d'erreur. La méthode retournera 1 si tout s'est bien passé, et 0 si une erreur est survenue.

```
int jouer (int ttint_plateau[N][N], int int_joueur, int int_x);
```

A gagné

À la fin de chaque tour de jeu, il faut vérifier si l'un des deux joueurs a gagné. Implémenter la fonction `aGagne` , qui retourne soit le numéro du joueur gagnant, soit 0 si c'est match nul, soit -1 si la partie n'est pas terminée.

```
int aGagne (int ttint_plateau[N][N]);
```

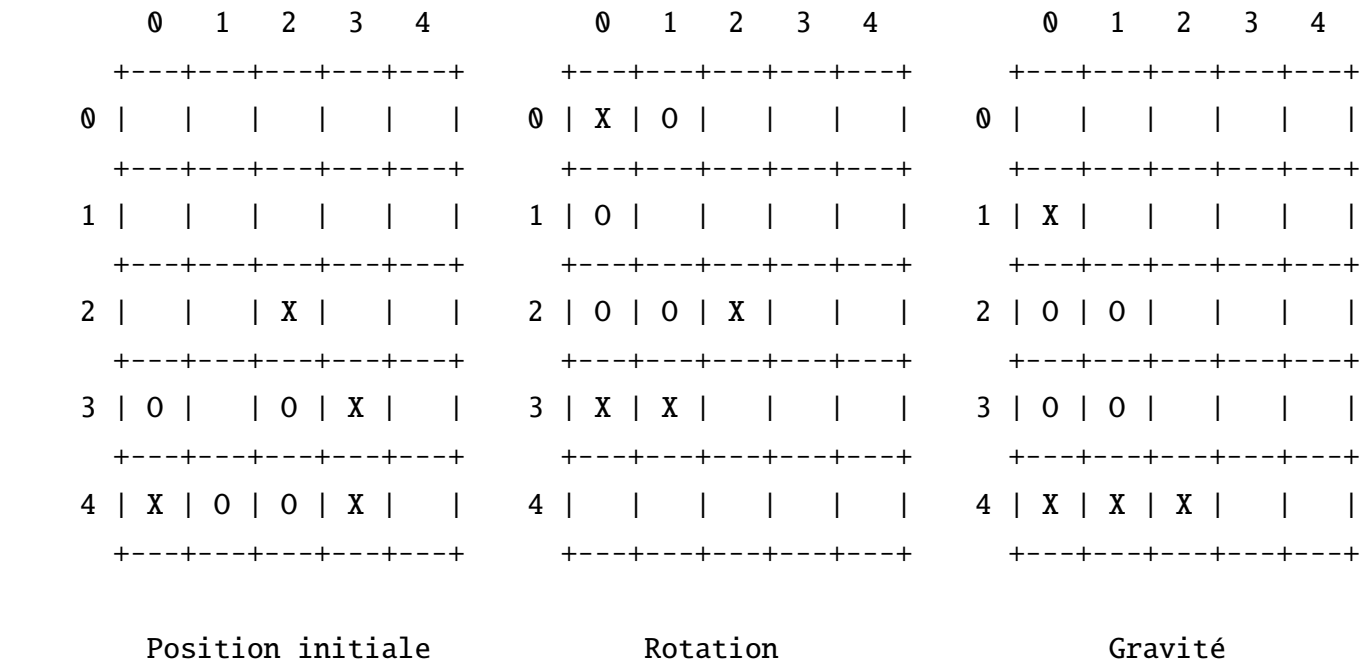
Tour de jeu

Écrire la méthode `tourDeJeu` qui permet de faire jouer les joueurs chacun leur tour, jusqu'à la fin de la partie. Lorsque la partie est finie, la grille complète devra être affichée, ainsi que le vainqueur. Lors d'une erreur de saisie de la part de l'utilisateur, le programme devra demander une nouvelle saisie, jusqu'à ce que l'utilisateur saisisse quelque chose de cohérent.

```
void tourDeJeu (int ttint_plateau[N][N]);
```

Variante du jeu (facultatif)

Afin de rendre le jeu de Puissance 4 un peu plus complexe, vous décidez de faire pivoter le plateau de 90 degrés dans le sens des aiguilles d'une montre. La gravité fera alors tomber les pièces dans une nouvelle position comme indiqué ci-dessous:



Afin que la condition de victoire ne soit pas trop dûe au hasard, les joueurs décident combien de fois chacun ils pourront faire pivoter le plateau. La rotation se fera toujours dans le sens des aiguilles d'une montre, et elle ne pourra se faire qu'à la fin du tour du joueur.

Ajouter les fonctions/procédures nécessaires pour prendre en compte cette variante dans le jeu de Puissance 4.