

Traces d'exécution

Les images que vous verrez à gauche sont les données de test que je modifie pour tester ce qui est demandé et à droite le résultat. Mais il se peut qu'il n'y est qu'une seul capture d'écran pour prouver la trace d'exécution

La procédure : initGrille

- Cas testé : grille non initialisé
- Résultat attendu : grille initialisé

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
```

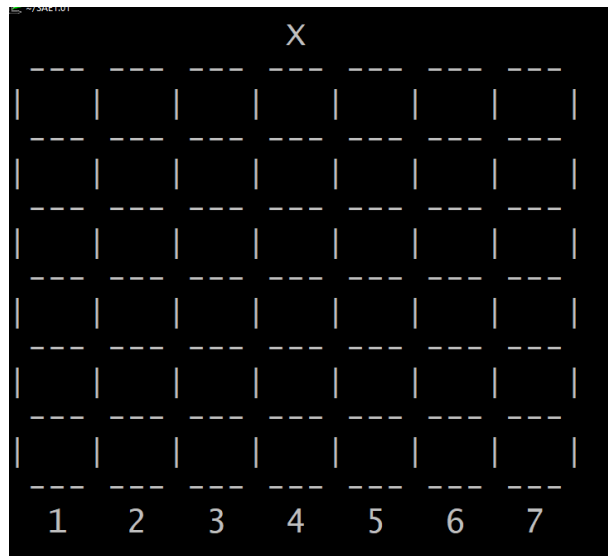


A 7x7 grid of dashed lines on a black background. At the top center, there is a white 'X'. At the bottom, the numbers 1 through 7 are written in white, centered under each column of the grid.

La procédure initGrille

- Cas testé : une grille déjà initialisée
- Résultat attendu : une grille initialisée

```
int main(){  
    t_grille g;  
    int colonne,ligne;  
    char vainqueur;  
    initGrille(g);  
  
    initGrille(g);  
    vainqueur=INCONNU;  
    afficher(g,PION_A,COLONNE_DEBUT);
```



La procédure : initGrille

- Cas testé : une grille déjà remplie avec des jetons
- Résultat attendu : grille initialisé

```
t_grille g;  
int colonne,ligne;  
char vainqueur;  
for(int i=0;i<4;i++){  
    g[i][i]=PION_A;  
}  
initGrille(g);
```



X							
1	2	3	4	5	6	7	

La procédure:afficher

- Cas testé : grille vide :
- Résultat attendu : grille affiché sans PION

```
int main(){  
    t_grille g;  
    int colonne,ligne;  
    char vainqueur;  
    initGrille(g);  
    vainqueur=INCONNU;  
    afficher(g,PION_A,COLONNE_DEBUT);  
}
```



X						
1	2	3	4	5	6	7

La procédure afficher

- Cas testé : grille remplie
- Résultat attendu : une grille avec 42 pions

```

3 int main(){
4     t_grille g;
5     int colonne,ligne;
6     char vainqueur;
7     initGrille(g);
8     for(int i=0;i<NBLIG;i++){
9         for(int j=0;j<NBCOL;j++){
10             g[i][j]=PION_A;
11         }
12     }
13     vainqueur=INCONNU;
14     afficher(g,PION_A,COLONNE_DEBUT);

```



	X						
	X	X	X	X	X	X	X
	X	X	X	X	X	X	X
	X	X	X	X	X	X	X
	X	X	X	X	X	X	X
	X	X	X	X	X	X	X
	X	X	X	X	X	X	X
	X	X	X	X	X	X	X
	X	X	X	X	X	X	X
1							
2							
3							
4							
5							
6							
7							

La procédure : afficher

- Cas à testé : Grille partiellement remplie
- Résultat attendu : grille affiché

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    g[5][0]=PION_A;
    g[4][0]=PION_B;
    g[5][1]=PION_A;
    g[5][2]=PION_B;
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
}
```



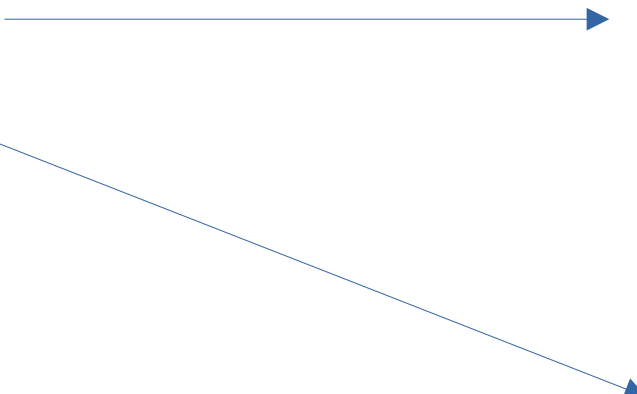
X						
---	---	---	---	---	---	---
---	---	---	---	---	---	---
---	---	---	---	---	---	---
---	---	---	---	---	---	---
	O					
---	---	---	---	---	---	---
	X		X		O	
---	---	---	---	---	---	---
1	2	3	4	5	6	7

La fonction : grillePleine

- Cas testé : grille non pleine
- Résultat attendu : false(0)

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,&colonne,&ligne);
        afficher(g,PION_B,COLONNE_DEBUT);
        printf("%d \n",grillepleine(g));
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);

    return EXIT_SUCCESS;
}
```



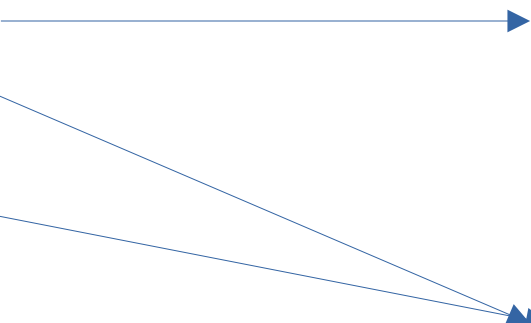
0						
				X		
	O			O		
	X			X		
	O	X	X	O		
1	2	3	4	5	6	7
0						

La fonction : grillePleine

- Cas testé : grille pleine
- Résultat attendu : true(1)

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,&colonne,&ligne);
        afficher(g,PION_B,COLONNE_DEBUT);
        printf("%d \n",grillepleine(g));
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            afficher(g,PION_A,COLONNE_DEBUT);
            printf("%d \n",grillepleine(g));
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);

    return EXIT_SUCCESS;
}
```



X						
X	X	O	X	O	X	O
O	X	O	X	O	X	O
O	X	O	O	X	O	X
X	O	X	O	X	O	X
X	O	X	O	X	O	X
O	X	O	X	O	X	O
1	2	3	4	5	6	7

La procédure : jouer

- Cas testé : jouer dans une colonne non pleine
- Résultat attendu : ligne et colonne conforme

```
int main()  
{  
    t_grille g;  
    int colonne,ligne;  
    char vainqueur;  
    initGrille(g);  
    vainqueur=INCONNU;  
    afficher(g,PION_A,COLONNE_DEBUT);  
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){  
        jouer(g,PION_A,&colonne,&ligne);  
        printf(" %d %d \n",ligne ,colonne);  
        afficher(g,PION_B,COLONNE_DEBUT);  
        if((estVainqueur(g,colonne,ligne))==1){  
            vainqueur=PION_A;  
        }  
        else if((grillepleine(g))==0){  
            jouer(g,PION_B,&colonne,&ligne);  
            printf(" %d %d \n",ligne ,colonne);  
            afficher(g,PION_A,COLONNE_DEBUT);  
            if((estVainqueur(g,colonne,ligne))==1){  
                vainqueur=PION_B;  
            }  
        }  
    }  
    finDePartie(vainqueur);  
    return EXIT_SUCCESS;  
}
```

Le 5(ligne) et le 2(colonne) sont les coordonnées par rapport au tableau
Car on compte le zéro dans le Pointage dans un tableau

Diagram illustrating the state of a 7x7 Connect Four board after a sequence of moves. The board is represented by a grid with columns numbered 1 to 7. The top row is labeled 'X'. The board state is as follows:

			O	X		

1 2 3 4 5 6 7

Décalez vous de colonne avec q pour la gauche et d pour la droite

5 2

0

Diagram illustrating the state of a 7x7 Connect Four board after a sequence of moves. The board is represented by a grid with columns numbered 1 to 7. The top row is labeled 'X'. The board state is as follows:

			X	O	X	

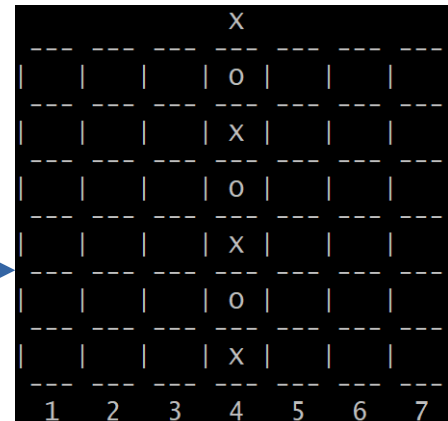
1 2 3 4 5 6 7

Décalez vous de colonne avec q pour la gauche et d pour la droite

La procédure jouer

- Cas testé : une colonne pleine
- Résultat attendu : invite à joueur dans une autre colonne

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,colonne,&ligne);
        afficher(g,PION_B,COLONNE_DEBUT);
        printf("%d \n",grillepleine(g));
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```



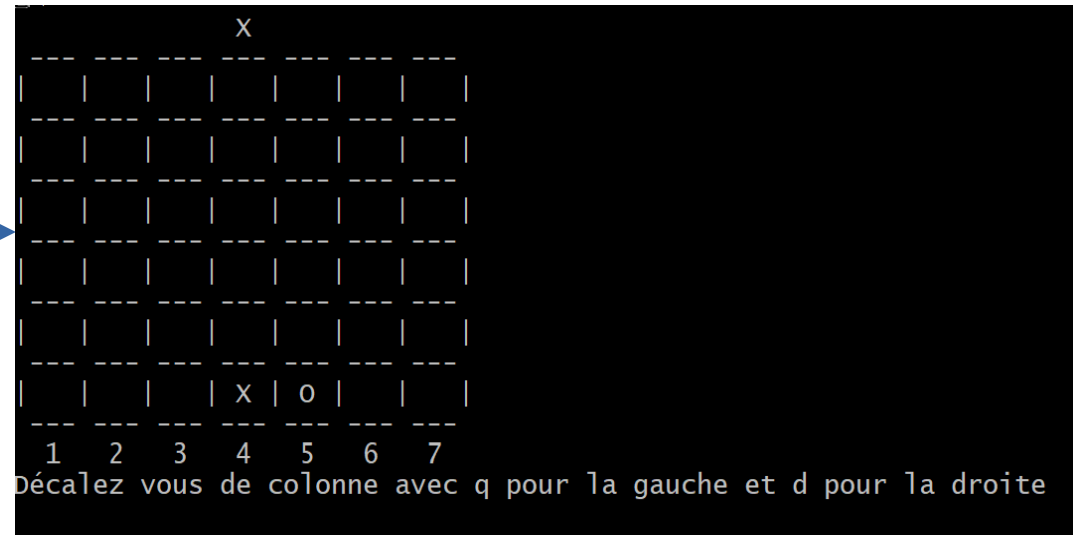
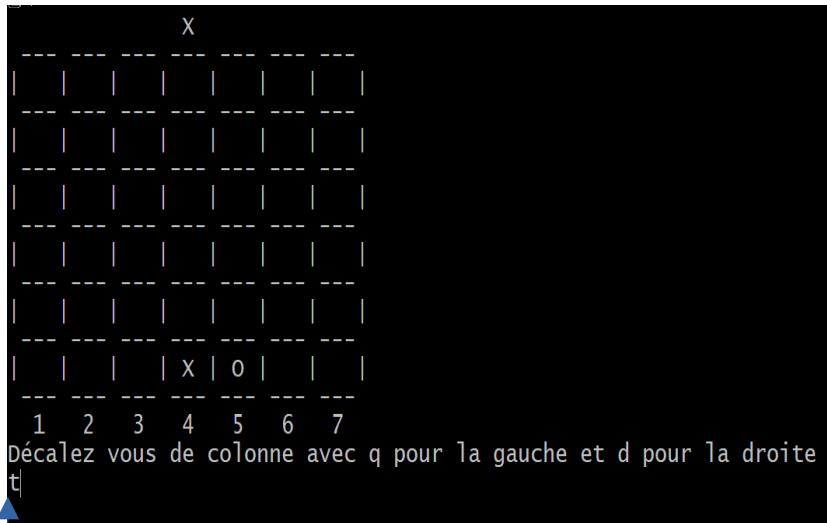
1 2 3 4 5 6 7

Décalez vous de colonne avec q pour la gauche et d pour la droite

erreur ligne complète il faut jouer dans une autre colonne

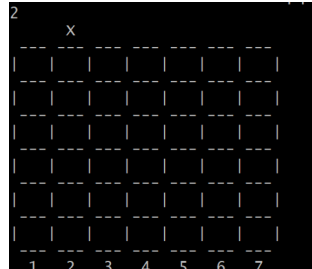
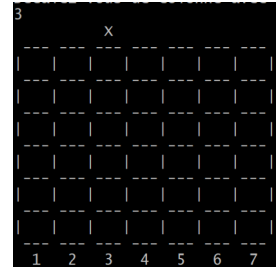
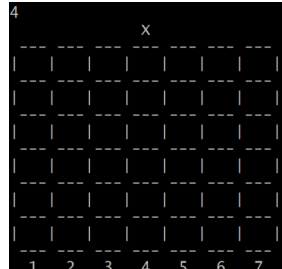
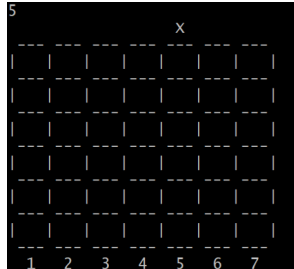
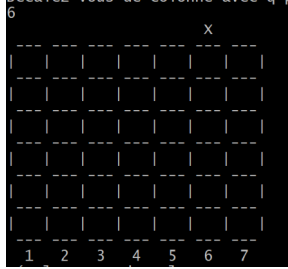
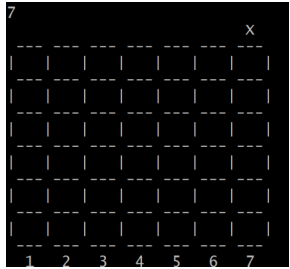
La fonction : ChoisirColonne

- Cas testé : saisir un caractère invalide
- Résultat attendu : redemande la saisie



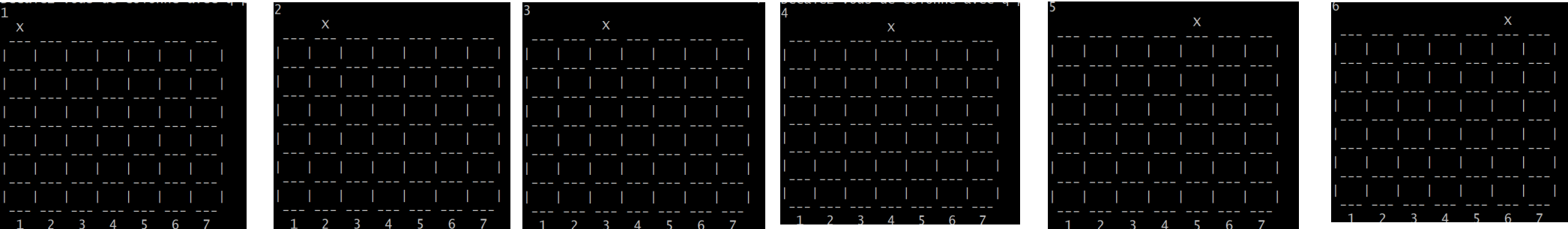
La fonction : ChoisirColonne

- Cas testé : colonne entre 2 et 7 et saisie de 'q '
- Résultat attendu : numéro de colonne qui décrémente



La fonction : ChoisirColonne

- Cas testé : colonne entre 1 et 6 et saisie de 'd'
- Résultat attendu : n° de colonne qui est incrémenté



Fonction : ChoisirColonne

- Cas testé : colonne de départ =7 et saisie de 'd'
- Résultat attendu : boucle en colonne 1

```
7
                                     x
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
1 2 3 4 5 6 7
Décalez vous de colonne avec q pour la gauche et d pour la droite
d
1
x
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
| | | | | | |
-----
1 2 3 4 5 6 7
Décalez vous de colonne avec q pour la gauche et d pour la droite
```


Fonction : ChoisirColonne

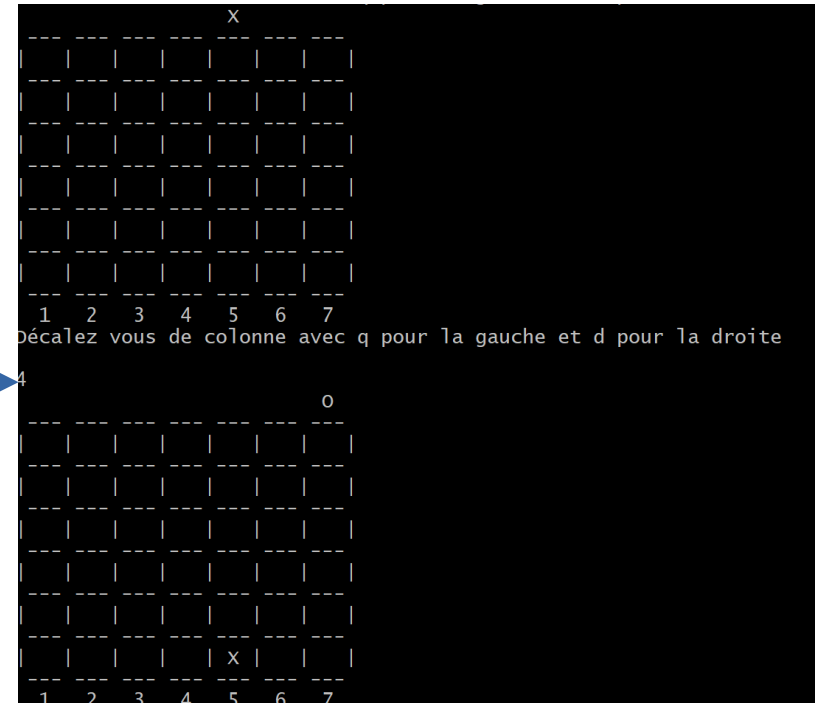
- Cas testé : colonne valide et saisie de la commande pour faire tomber le pion
- Résultat attendu : le numéro de colonne courant :

Numéro de colonne
et il ne faut pas oublier que

Je compte le zéro c'est pour cela que

Avoir choisi la colonne 5 retourne 4

C'est pour bien placer le pion dans le tableau



Fonction : TrouverLigne

- Cas testé : colonne pleine
- Résultat attendu : -1

```
Décalez vous de colonne avec q pour la gauche et d pour la droite
X
---
|  |  |  |  |  |  | o |
---
|  |  |  |  |  |  | x |
---
|  |  |  |  |  |  | o |
---
|  |  |  |  |  |  | x |
---
|  |  |  |  |  |  | o |
---
|  |  |  |  |  |  | x |
---
  1  2  3  4  5  6  7
Décalez vous de colonne avec q pour la gauche et d pour la droite
erreur ligne complète il faut jouer dans une autre colonne
-1
```

Fonction : TrouverLigne

- Cas testé : colonne non pleine
- Résultat attendu : numéro de la ligne

Numéro de ligne en
comptant le zéro
Donc 0 à 5

[illegible]

Fonction : EstVainqueur

- Cas testé : Pas 4 pions identiques alignés
- Résultat attendu: False(0)

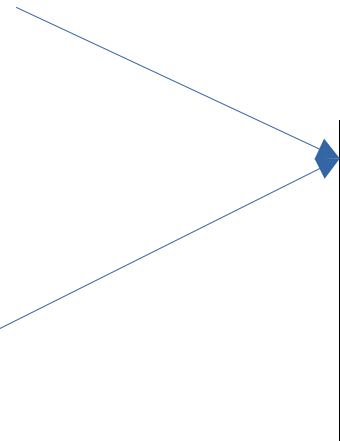
```
int main(){
    t_grille g;
    int colonne, ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g, PION_A, COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g, PION_A, &colonne, &ligne);
        printf("%d\n", estVainqueur(g, colonne, ligne));
        afficher(g, PION_B, COLONNE_DEBUT);
        if((estVainqueur(g, colonne, ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g, PION_B, &colonne, &ligne);
            printf("%d\n", estVainqueur(g, colonne, ligne));
            afficher(g, PION_A, COLONNE_DEBUT);
            if((estVainqueur(g, colonne, ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```

0							
	0	X	0	X			
	1	2	3	4	5	6	7

Fonction : EstVainqueur

- Cas attendu : colonne d'au moins 4 pions
- Résultat attendu : True(1)

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,&colonne,&ligne);
        printf("%d\n",estVainqueur(g,colonne,ligne));
        afficher(g,PION_B,COLONNE_DEBUT);
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            printf("%d\n",estVainqueur(g,colonne,ligne));
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```



1							
			X				
			X				
		O	X	O			
O	X	O	X	X	O		
1	2	3	4	5	6	7	

Le vainqueur de la partie est x (joueur 1)

Fonction : EstVainqueur

- Cas testé : une ligne de 4 pions
- Résultat attendu : True(1)

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur!=INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,&colonne,&ligne);
        printf("%d\n",estVainqueur(g,colonne,ligne));
        afficher(g,PION_B,COLONNE_DEBUT);
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            printf("%d\n",estVainqueur(g,colonne,ligne));
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```

1

				X		
			X			
			X	X		
O	O	O	O	X		

1 2 3 4 5 6 7

le vainqueur de la partie est O (joueur 2)

Fonction : EstVainqueur

- Cas testé : diagonale descendante
- Résultat attendu : True(1)

```

E ~/SAE1.01
      0
  -----
  | | | | | | |
  -----
  | | | | | | |
  -----
  | | | | | | |
  -----
  | | x | o | | | |
  -----
  | | x | x | o | x | |
  -----
  | | x | o | x | o | o |
  -----
  1 2 3 4 5 6 7
Décalez vous de colonne avec q pour la gauche et d pour la droite
1
      x
  -----
  | | | | | | |
  -----
  | | | | | | |
  -----
  | | o | | | | |
  -----
  | | x | o | | | |
  -----
  | | x | x | o | x | |
  -----
  | | x | o | x | o | o |
  -----
  1 2 3 4 5 6 7
le vainqueur de la partie est o (joueur 2)
martin@Lucas ~/SAE1.01
$

```

Fonction : EstVainqueur

- Cas testé : Diagonale ascendante
- Résultat attendu : True(1)

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&&((grillepleine(g))==0)){
        jouer(g,PION_A,&colonne,&ligne);
        printf("%d\n",estVainqueur(g,colonne,ligne));
        afficher(g,PION_B,COLONNE_DEBUT);
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            printf("%d\n",estVainqueur(g,colonne,ligne));
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```

```

      X
-----
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   | O |   |   | X | X |
|   | X |   | X | O | O |
|   | O |   | O | O | X |
1 2 3 4 5 6 7
Décalez vous de colonne avec q pour la gauche et d pour la droite
1
      O
-----
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   | O |   |   | X | X |
|   | O |   | X | O | O |
|   | O | X | O | O | X |
1 2 3 4 5 6 7
Le vainqueur de la partie est X (joueur 1)
martin@Lucas ~/SAE1.01
$
```


Fonction : EstVainqueur

- Cas testé : ligne d'au moins 4 pions interrompue par un pion adverse
- Résultat : False(0)

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,&colonne,&ligne);
        printf("%d\n",estVainqueur(g,colonne,ligne));
        afficher(g,PION_B,COLONNE_DEBUT);
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            printf("%d\n",estVainqueur(g,colonne,ligne));
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```

Terminal 1 (Top):

```

$ ./SAE1.01
Décalez vous de colonne avec q pour la gauche et d pour la droite
x
-----
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | o | | |
| | | x | o | x |
-----
1 2 3 4 5 6 7

```

Terminal 2 (Bottom):

```

0
Décalez vous de colonne avec q pour la gauche et d pour la droite
o
-----
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | o | | |
| | | x | o | x |
-----
1 2 3 4 5 6 7

```

procédure : finDePartie

- Cas testé : pion en entrée = PION_A
- Résultat attendu : affiche X vainqueur

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,colonne,&ligne);
        printf("%d\n",estVainqueur(g,colonne,ligne));
        afficher(g,PION_B,COLONNE_DEBUT);
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,colonne,&ligne);
            printf("%d\n",estVainqueur(g,colonne,ligne));
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```



0						
			O			
		O	O			
X	X	X	X			
1	2	3	4	5	6	7

Le vainqueur de la partie est x (joueur 1)

Procédure : finDePartie

- Cas testé : pion entré = PION_b
- Resultat attendu : affiche O vainqueur

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,&colonne,&ligne);
        printf("%d\n",estVainqueur(g,colonne,ligne));
        afficher(g,PION_B,COLONNE_DEBUT);
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            printf("%d\n",estVainqueur(g,colonne,ligne));
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```

le vainqueur de la partie est O (joueur 2)

Procedure : finDePartie

- Cas testé : pion en entrée = Vide
- Résultat attendu : affiche Match NUL

```
int main(){
    t_grille g;
    int colonne,ligne;
    char vainqueur;
    initGrille(g);
    vainqueur=INCONNU;
    afficher(g,PION_A,COLONNE_DEBUT);
    while((vainqueur==INCONNU)&((grillepleine(g))==0)){
        jouer(g,PION_A,&colonne,&ligne);
        printf("%d\n",estVainqueur(g,colonne,ligne));
        afficher(g,PION_B,COLONNE_DEBUT);
        if((estVainqueur(g,colonne,ligne))==1){
            vainqueur=PION_A;
        }
        else if((grillepleine(g))==0){
            jouer(g,PION_B,&colonne,&ligne);
            printf("%d\n",estVainqueur(g,colonne,ligne));
            afficher(g,PION_A,COLONNE_DEBUT);
            if((estVainqueur(g,colonne,ligne))==1){
                vainqueur=PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    return EXIT_SUCCESS;
}
```



	X						
	X	O	X	O	X	X	O
	O	X	O	X	O	X	O
	X	O	X	O	O	X	O
	X	O	X	O	X	O	X
	X	O	X	X	O	X	O
	O	X	O	X	O	X	O
	1	2	3	4	5	6	7

Match nul