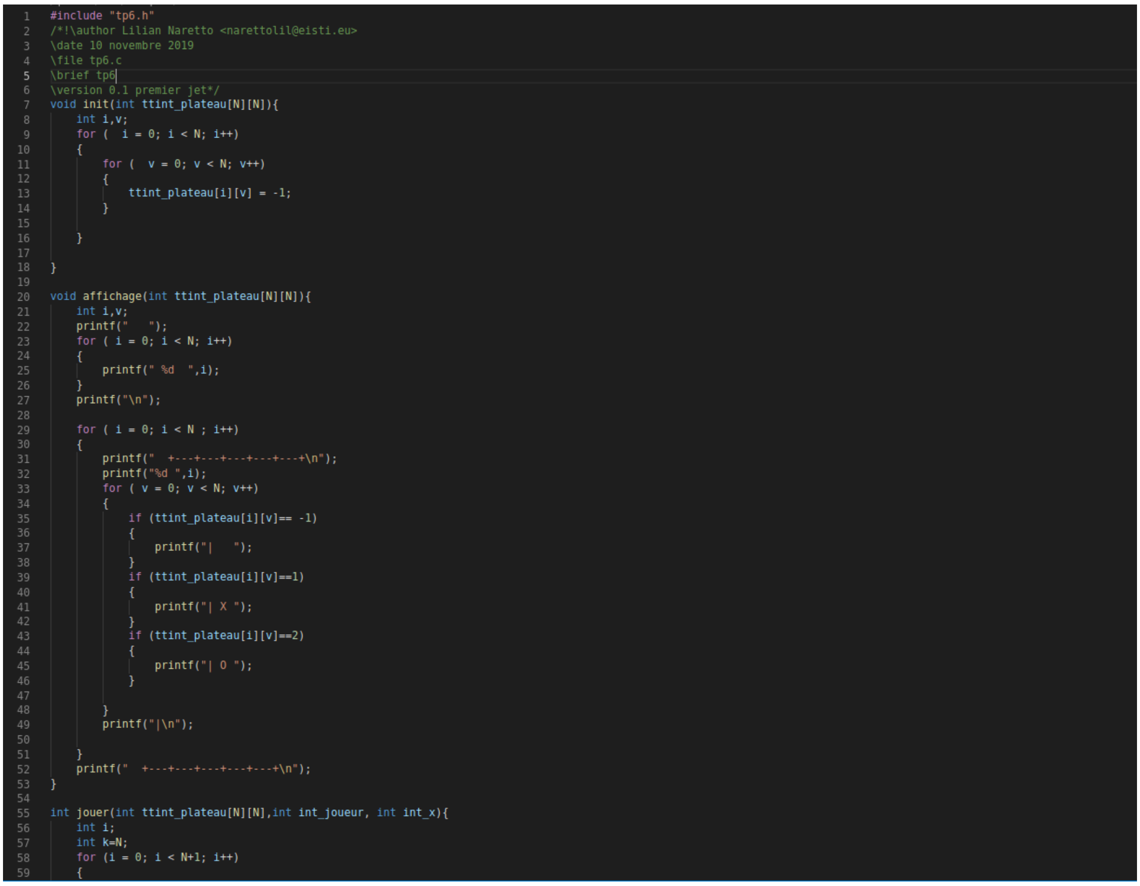
PROGRAMMATION PROCEDURALE - Année 2019/2020

CLASSE CYCLE D'INGENIEUR ING1G04

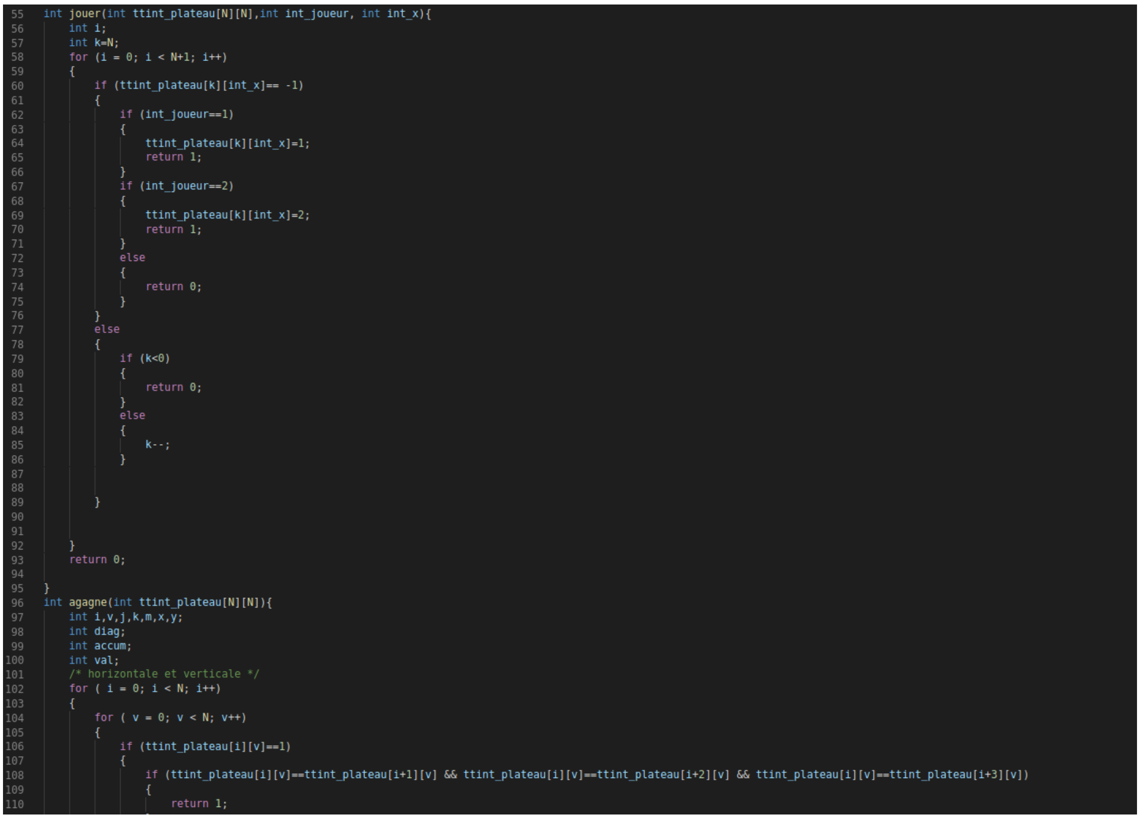
INTRODUCTION AU C :

TP6

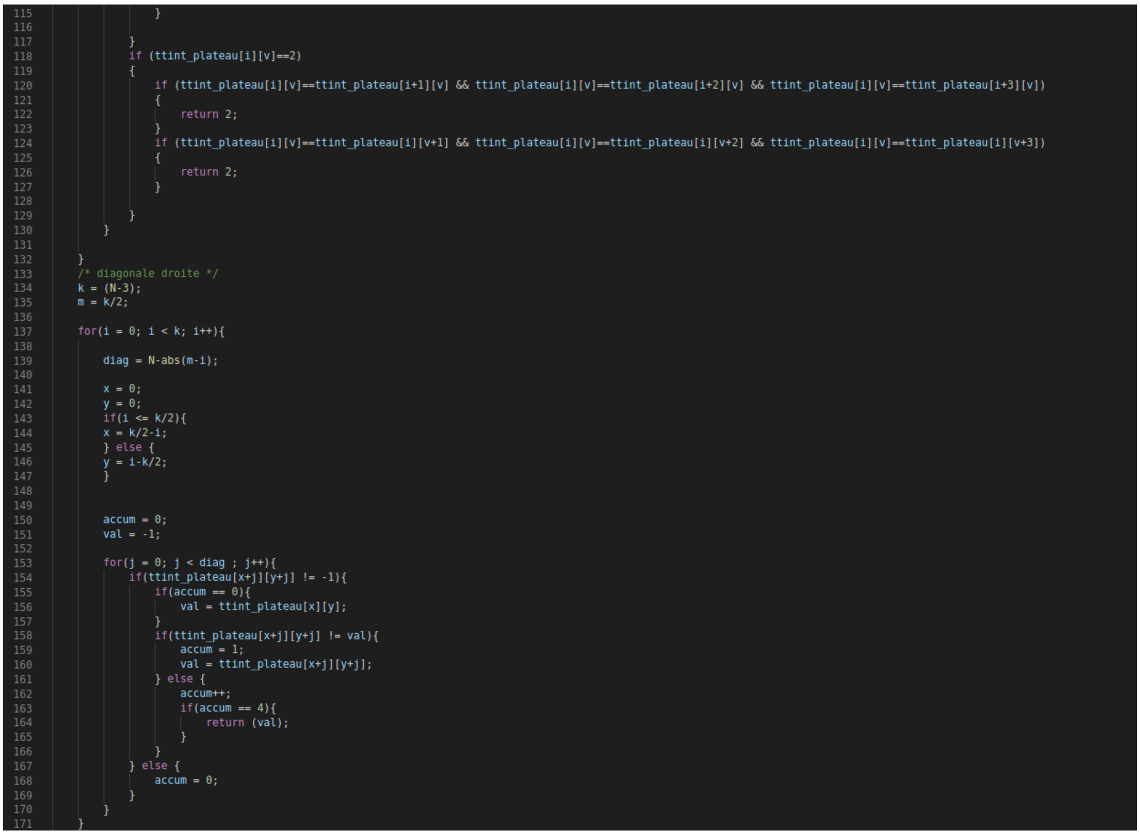
Made by :
Lilian Naretto



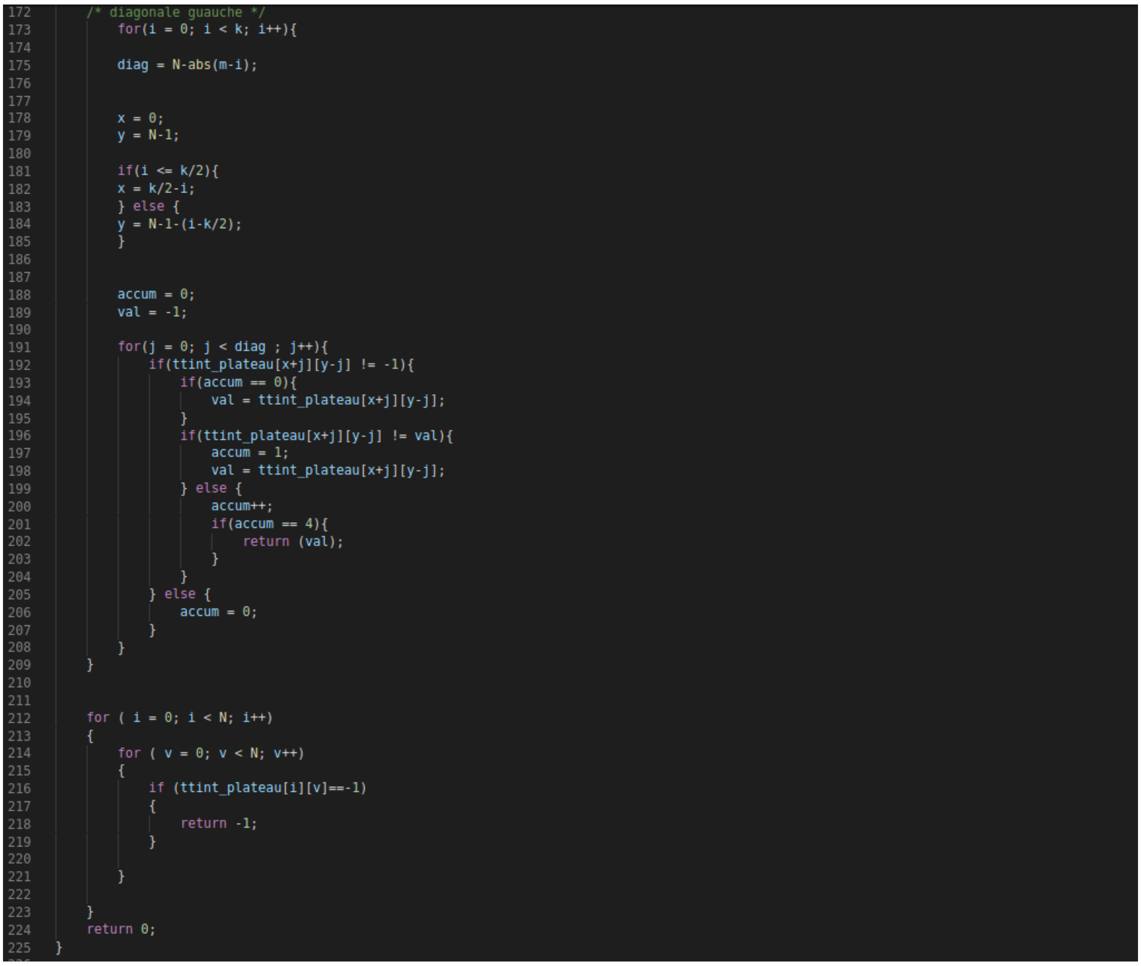
```c
#include "tp6.h"
/*!\author Lilian Naretto <narettolil@eisti.eu>
\date 10 novembre 2019
\file tp6.c
\brief tp6
\version 0.1 premier jet*/
void init(int ttint_plateau[N][N]){
    int i,v;
    for (  i = 0; i < N; i++)
    {
        for (  v = 0; v < N; v++)
        {
            ttint_plateau[i][v] = -1;
        }

    }

}

void affichage(int ttint_plateau[N][N]){
    int i,v;
    printf("    ");
    for ( i = 0; i < N; i++)
    {
        printf(" %d  ",i);
    }
    printf("\n");

    for ( i = 0; i < N ; i++)
    {
        printf("  +---+---+---+---+---+\n");
        printf("%d ",i);
        for ( v = 0; v < N; v++)
        {
            if (ttint_plateau[i][v]== -1)
            {
                printf("|   ");
            }
            if (ttint_plateau[i][v]==1)
            {
                printf("| X ");
            }
            if (ttint_plateau[i][v]==2)
            {
                printf("| O ");
            }

        }
        printf("|\n");

    }
    printf("  +---+---+---+---+---+\n");
}

int jouer(int ttint_plateau[N][N],int int_joueur, int int_x){
    int i;
    int k=N;
    for (i = 0; i < N+1; i++)
    {
```



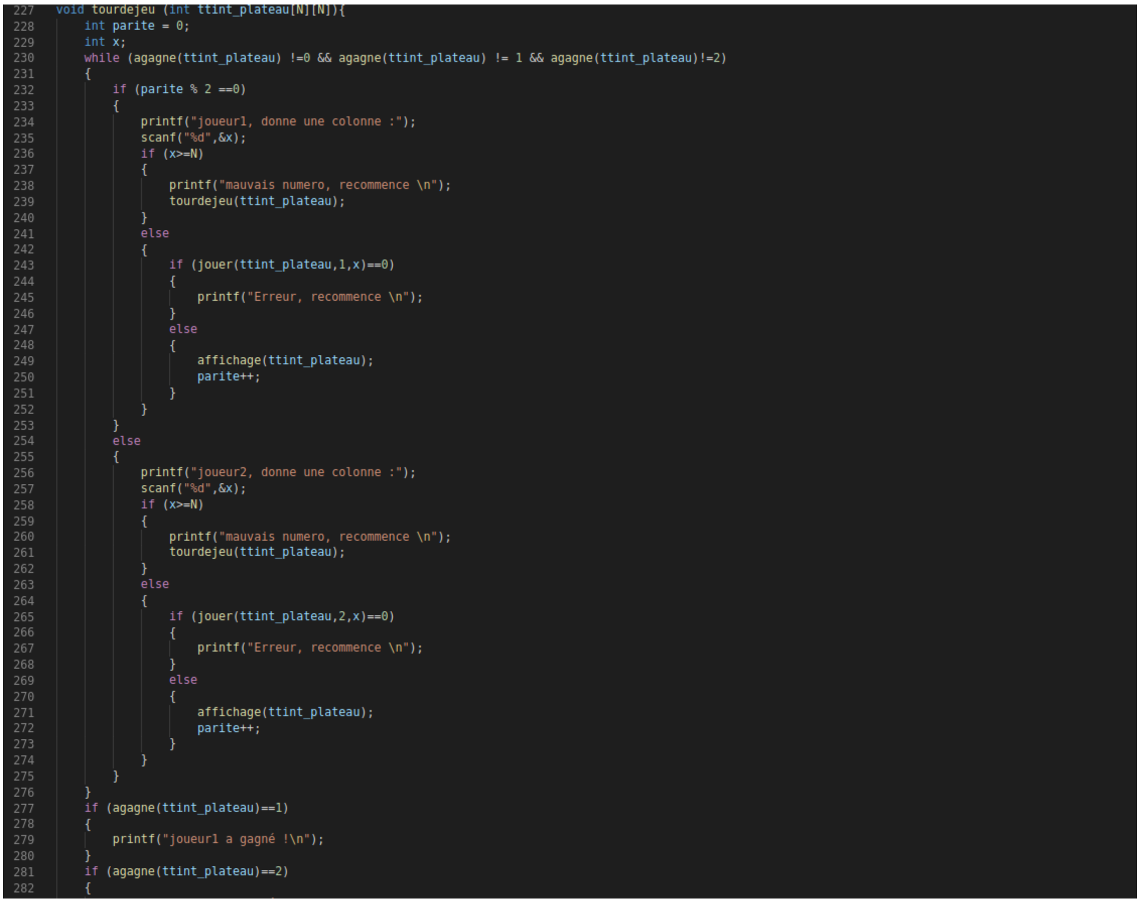
```c
int jouer(int ttint_plateau[N][N],int int_joueur, int int_x){
    int i;
    int k=N;
    for (i = 0; i < N+1; i++)
    {
        if (ttint_plateau[k][int_x]== -1)
        {
            if (int_joueur==1)
            {
                ttint_plateau[k][int_x]=1;
                return 1;
            }
            if (int_joueur==2)
            {
                ttint_plateau[k][int_x]=2;
                return 1;
            }
            else
            {
                return 0;
            }
        }
        else
        {
            if (k<0)
            {
                return 0;
            }
            else
            {
                k--;
            }


        }


    }
    return 0;

}
int agagne(int ttint_plateau[N][N]){
    int i,v,j,k,m,x,y;
    int diag;
    int accum;
    int val;
    /* horizontale et verticale */
    for ( i = 0; i < N; i++)
    {
        for ( v = 0; v < N; v++)
        {
            if (ttint_plateau[i][v]==1)
            {
                if (ttint_plateau[i][v]==ttint_plateau[i+1][v] && ttint_plateau[i][v]==ttint_plateau[i+2][v] && ttint_plateau[i][v]==ttint_plateau[i+3][v])
                {
                    return 1;
```



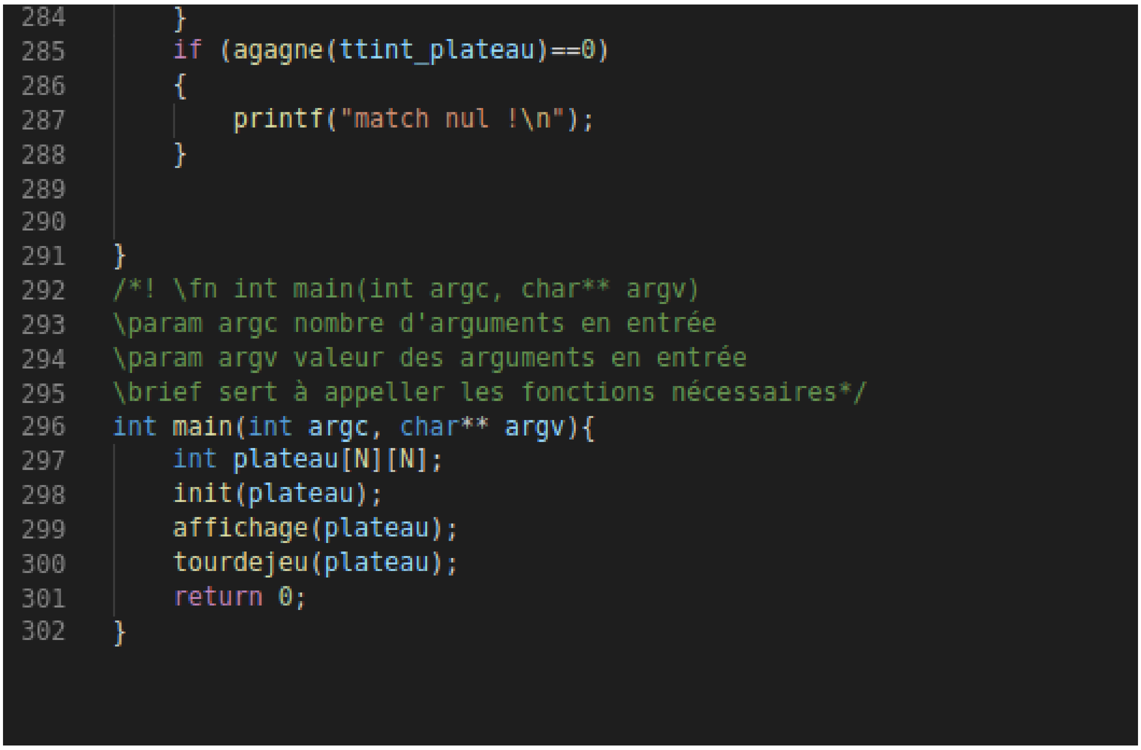
```
                }

            }
            if (ttint_plateau[i][v]==2)
            {
                if (ttint_plateau[i][v]==ttint_plateau[i+1][v] && ttint_plateau[i][v]==ttint_plateau[i+2][v] && ttint_plateau[i][v]==ttint_plateau[i+3][v])
                {
                    return 2;
                }
                if (ttint_plateau[i][v]==ttint_plateau[i][v+1] && ttint_plateau[i][v]==ttint_plateau[i][v+2] && ttint_plateau[i][v]==ttint_plateau[i][v+3])
                {
                    return 2;
                }

            }
        }

    }
    /* diagonale droite */
    k = (N-3);
    m = k/2;

    for(i = 0; i < k; i++){

        diag = N-abs(m-i);

        x = 0;
        y = 0;
        if(i <= k/2){
        x = k/2-i;
        } else {
        y = i-k/2;
        }


        accum = 0;
        val = -1;

        for(j = 0; j < diag ; j++){
            if(ttint_plateau[x+j][y+j] != -1){
                if(accum == 0){
                    val = ttint_plateau[x][y];
                }
                if(ttint_plateau[x+j][y+j] != val){
                    accum = 1;
                    val = ttint_plateau[x+j][y+j];
                } else {
                    accum++;
                    if(accum == 4){
                        return (val);
                    }
                }
            } else {
                accum = 0;
            }
        }
    }
```



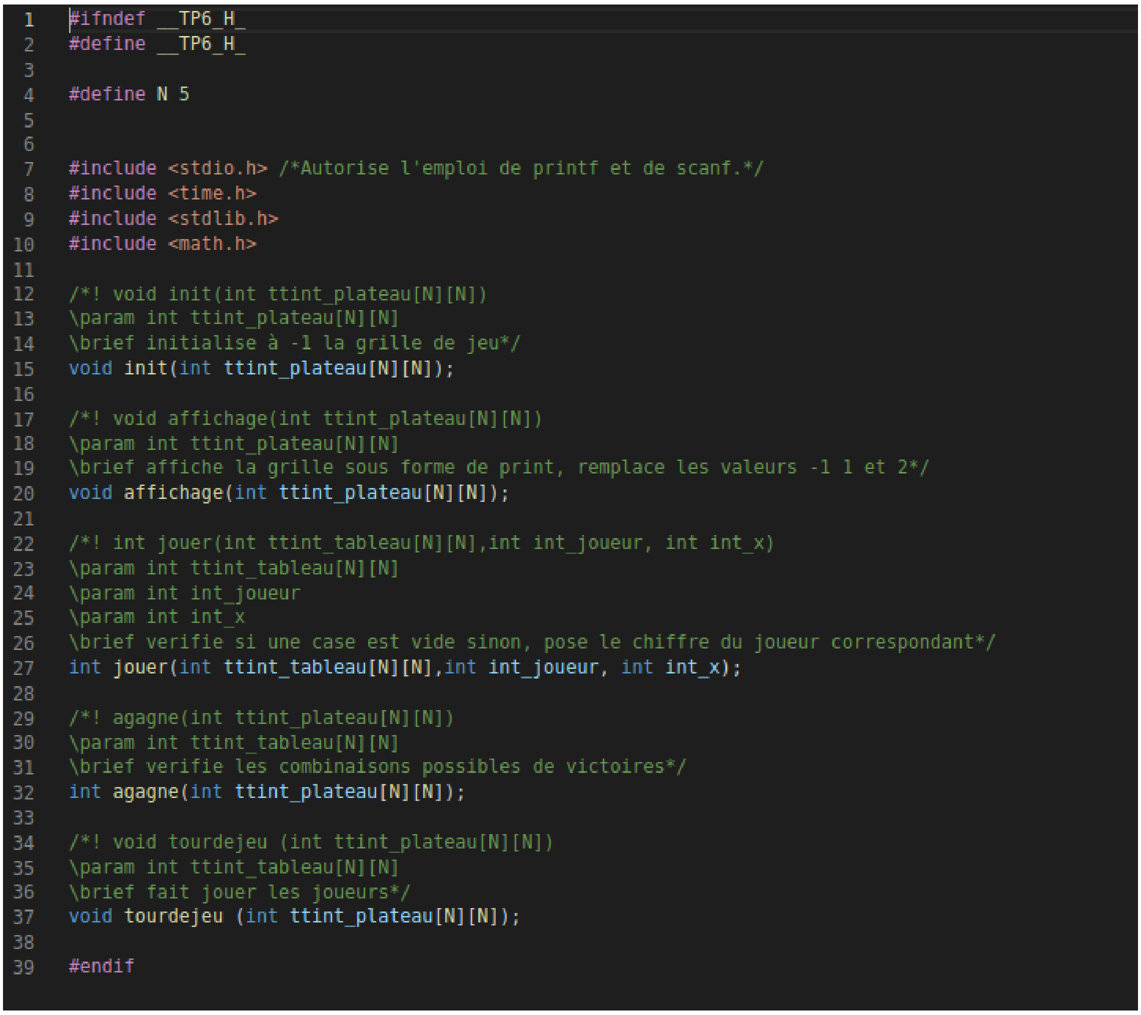
```
    /* diagonale guauche */
        for(i = 0; i < k; i++){

        diag = N-abs(m-i);


        x = 0;
        y = N-1;

        if(i <= k/2){
        x = k/2-i;
        } else {
        y = N-1-(i-k/2);
        }


        accum = 0;
        val = -1;

        for(j = 0; j < diag ; j++){
            if(ttint_plateau[x+j][y-j] != -1){
                if(accum == 0){
                    val = ttint_plateau[x+j][y-j];
                }
                if(ttint_plateau[x+j][y-j] != val){
                    accum = 1;
                    val = ttint_plateau[x+j][y-j];
                } else {
                    accum++;
                    if(accum == 4){
                        return (val);
                    }
                }
            } else {
                accum = 0;
            }
        }
    }


    for ( i = 0; i < N; i++)
    {
        for ( v = 0; v < N; v++)
        {
            if (ttint_plateau[i][v]==-1)
            {
                return -1;
            }

        }

    }
    return 0;
}
```



```c
void tourdejeu (int ttint_plateau[N][N]){
    int parite = 0;
    int x;
    while (agagne(ttint_plateau) !=0 && agagne(ttint_plateau) != 1 && agagne(ttint_plateau)!=2)
    {
        if (parite % 2 ==0)
        {
            printf("joueur1, donne une colonne :");
            scanf("%d",&x);
            if (x>=N)
            {
                printf("mauvais numero, recommence \n");
                tourdejeu(ttint_plateau);
            }
            else
            {
                if (jouer(ttint_plateau,1,x)==0)
                {
                    printf("Erreur, recommence \n");
                }
                else
                {
                    affichage(ttint_plateau);
                    parite++;
                }
            }
        }
        else
        {
            printf("joueur2, donne une colonne :");
            scanf("%d",&x);
            if (x>=N)
            {
                printf("mauvais numero, recommence \n");
                tourdejeu(ttint_plateau);
            }
            else
            {
                if (jouer(ttint_plateau,2,x)==0)
                {
                    printf("Erreur, recommence \n");
                }
                else
                {
                    affichage(ttint_plateau);
                    parite++;
                }
            }
        }
    }
    if (agagne(ttint_plateau)==1)
    {
        printf("joueur1 a gagné !\n");
    }
    if (agagne(ttint_plateau)==2)
    {
```



```c
    }
    if (agagne(ttint_plateau)==0)
    {
        printf("match nul !\n");
    }


}
/*! \fn int main(int argc, char** argv)
\param argc nombre d'arguments en entrée
\param argv valeur des arguments en entrée
\brief sert à appeller les fonctions nécessaires*/
int main(int argc, char** argv){
    int plateau[N][N];
    init(plateau);
    affichage(plateau);
    tourdejeu(plateau);
    return 0;
}
```



```c
#ifndef __TP6_H_
#define __TP6_H_

#define N 5


#include <stdio.h> /*Autorise l'emploi de printf et de scanf.*/
#include <time.h>
#include <stdlib.h>
#include <math.h>

/*! void init(int ttint_plateau[N][N])
\param int ttint_plateau[N][N]
\brief initialise à -1 la grille de jeu*/
void init(int ttint_plateau[N][N]);

/*! void affichage(int ttint_plateau[N][N])
\param int ttint_plateau[N][N]
\brief affiche la grille sous forme de print, remplace les valeurs -1 1 et 2*/
void affichage(int ttint_plateau[N][N]);

/*! int jouer(int ttint_tableau[N][N],int int_joueur, int int_x)
\param int ttint_tableau[N][N]
\param int int_joueur
\param int int_x
\brief verifie si une case est vide sinon, pose le chiffre du joueur correspondant*/
int jouer(int ttint_tableau[N][N],int int_joueur, int int_x);

/*! agagne(int ttint_plateau[N][N])
\param int ttint_tableau[N][N]
\brief verifie les combinaisons possibles de victoires*/
int agagne(int ttint_plateau[N][N]);

/*! void tourdejeu (int ttint_plateau[N][N])
\param int ttint_tableau[N][N]
\brief fait jouer les joueurs*/
void tourdejeu (int ttint_plateau[N][N]);

#endif
```