```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define TAILLE 9 // taille de la grille

typedef int tGrille[TAILLE][TAILLE]; // Définition du type pour la grille de
sudoku
tGrille grille1; // Grille de sudoku globale
int numColonne, numLigne, valeur; // Variables globales pour les indices de
colonne, ligne et valeur

void chargerGrille(tGrille g);
void afficherGrille(tGrille g);
int grillePleine(tGrille g);
void saisir(int *S, int n);
bool possible(tGrille g, int ligne, int colonne, int valeur);

int main(){
    chargerGrille(grille1); // Chargement de la grille depuis un fichier
    while(!grillePleine(grille1)){ // Boucle principale du jeu jusqu'à ce que la
grille soit pleine
        afficherGrille(grille1); // Affichage de la grille
        printf("Indices de la case ? \n");
        saisir(&numLigne, TAILLE); // Saisie de l'indice de la ligne
        saisir(&numColonne, TAILLE); // Saisie de l'indice de la colonne
        numLigne--; //décrémentation de l'indice de la ligne
        numColonne--; //décrémentation de l'indice de la colonne
        if(grille1[numLigne][numColonne]!=0){ // Vérifie si la case n'est pas
déjà remplie
            printf("IMPOSSIBLE, la case n'est pas libre. \n");
        } else {
            printf("Valeur a inserer? \n");
            saisir(&valeur, TAILLE); // Saisie de la valeur à insérer
            if(possible(grille1, numLigne, numColonne, valeur)){ // Vérifie si
la valeur peut être insérée dans la grille
                grille1[numLigne][numColonne] = valeur; // Insertion de la
valeur dans la grille
            }
        }
    }
    printf("Grille pleine, fin de partie\n");
    return 0;
}

// Charge une grille de sudoku depuis un fichier
void chargerGrille(tGrille g){
    char nomFichier[30]; // Nom du fichier à charger
    FILE * f;

    printf("Nom du fichier ? ");
    scanf("%s", nomFichier);

    f = fopen(nomFichier, "rb"); // Ouverture du fichier
    if (f==NULL){ // Vérifie si le fichier est ouvert
        printf("\n ERREUR sur le fichier %s\n", nomFichier); // Affichage d'un
message d'erreur si le fichier n'est pas ouvert
    } else {
        fread(g, sizeof(int), TAILLE*TAILLE, f); // Lecture de la grille depuis
le fichier
    }
    fclose(f); // Fermeture du fichier
}
```

```
// Affiche la grille de sudoku
void afficherGrille(tGrille g){
    printf("    1  2  3   4  5  6   7  8  9\n");
    printf("  +---------+---------+---------+\n");
    int i, j, num = 1;
    for (i = 0; i < TAILLE; i++) {
        printf("%d |", num);
        num++;
        for(j=0; j<TAILLE; j++){
            if(g[i][j] == 0){
                printf(" . "); // Affiche '.' si la case est vide
            } else {
                printf(" %d ", g[i][j]); // Affiche la valeur de la case
            }
            if ((j+1) % 3 == 0){
                printf("|"); // Séparateur de blocs
            }
            if (j == 8 && (i == 2 || i == 5)){
                printf("\n  +---------+---------+---------+"); // Lignes de
séparation entre blocs
            }
        }
        printf("\n");
    }
    printf("  +---------+---------+---------+\n");
}

// Fonction pour vérifier si la grille est pleine
int grillePleine(tGrille g){
    int i, j;
    for (i = 0; i < TAILLE; i++) {
        for(j=0; j<TAILLE; j++){
            if(g[i][j] == 0){
                return 0; // La grille n'est pas pleine
            }
        }
    }
    return 1; // La grille est pleine
}

// Procédure saisie pour saisir une case et entrer une valeur
void saisir(int *S, int n) {
    int valeur;
    char input[50]; // Chaîne de caractères pour stocker l'entrée utilisateur
    do {
        scanf("%s", input);
        // Vérification si la saisie peut être convertie en entier
        if (sscanf(input, "%d", &valeur) != 0) {
            // Si la valeur est dans la plage valide (entre 1 et n)
            if (valeur >= 1 && valeur <= n) {
                *S = valeur ; // Affectation de la valeur saisie
                break; // Sortie de la boucle si la valeur est valide
            } else {
                printf("La valeur doit etre entre 1 et %d.\n", n);
            }
        } else {
            printf("Veuillez saisir un entier valide.\n");
        }
    } while (1); // Boucle tant que la valeur saisie n'est pas valide
}

// Vérifie si une valeur peut être insérée dans une case sans qu'il y ai de
doublons dans la colonne, la ligne ou le bloc
bool possible(tGrille g, int ligne, int colonne, int valeur) {
```

```
    // Vérification de la ligne
    for (int i = 0; i < TAILLE; i++) {
        if (g[ligne][i] == valeur && i != colonne) {
            printf("Impossible, la valeur %d est deja presente dans la ligne.\
n", valeur);
            return false; // Doublon dans la ligne
        }
    }

    // Vérification de la colonne
    for (int i = 0; i < TAILLE; i++) {
        if (g[i][colonne] == valeur && i != ligne) {
            printf("Impossible, la valeur %d est deja presente dans la colonne.\
n", valeur);
            return false; // Doublon dans la colonne
        }
    }

    // Calcul des limites du bloc
    int debutLigne = (ligne / 3) * 3;
    int debutColonne = (colonne / 3) * 3;
    int finLigne = debutLigne + 3;
    int finColonne = debutColonne + 3;

    // Vérification du bloc
    for (int i = debutLigne; i < finLigne; i++) {
        for (int j = debutColonne; j < finColonne; j++) {
            if (g[i][j] == valeur && (i != ligne || j != colonne)) {
                printf("Impossible, la valeur %d est deja presente dans le
bloc.\n", valeur);
                return false; // Doublon dans le bloc
            }
        }
    }

    return true; // Aucun doublon trouvé
}
```