# Deuxième exercice : - Sudoku

Pour découvrir le jeu: https://sudoku.com/fr

Description :

Le sudoku est un puzzle utilisant une grille carrée composée de 9 régions (représentées en gras sur la figure 1) ; chaque région est elle-même composée de 9 cases. On part d'une grille partiellement remplie et on doit la compléter en respectant les trois règles suivantes :

- Chaque colonne doit contenir les 9 chiffres de 1 à 9
- Chaque ligne doit contenir les 9 chiffres de 1 à 9
- Chaque région doit contenir les 9 chiffres de 1 à 9

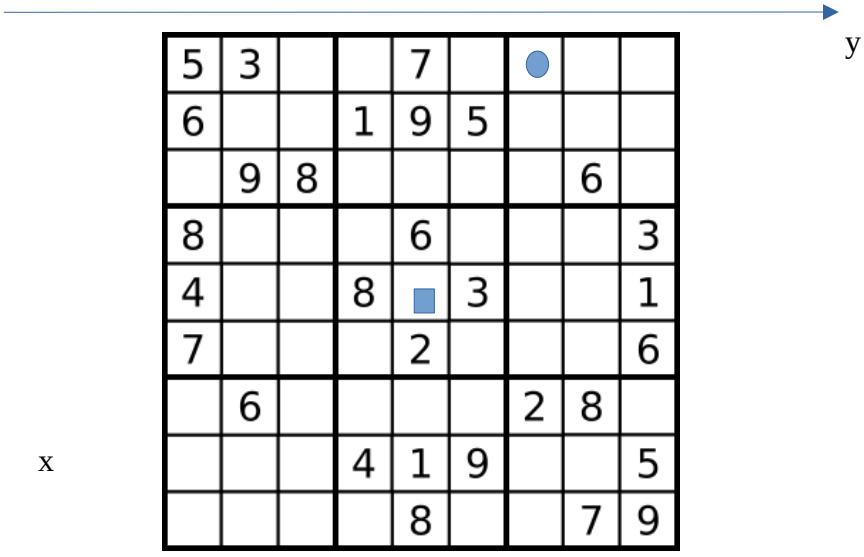
Une grille de sudoku est représentée par un tableau de unsigned char de 9 lignes et 9 colonnes :

**typedef uint8_t sudoku[9[9];**

**Ici on utilise des uint8_t pour des raisons de place mémoire.**

Ainsi on pourra déclarer une variable s de type sudoku :
**sudoku s;** // ce qui est équivalent à **uint8_t s[9][9]**;

Une case vide prendra la valeur entière 0, une case remplie la valeur entière correspondante (de 1 à 9)




Exemple de grille de sudoku.

Dans cet exemple s[0][0] vaut 5, s[2][7] vaut 6, s[0][3] vaut 0, etc.

Si l'on considère la case vide où on a dessiné un petit rectangle, il n'est pas possible de poser 4,8,3,1 à cause de la ligne, et 7,9,6,2,1,8 à cause de la colonne. Il n'y a donc qu'**une** seule possibilité. (la valeur 5)

Si l'on considère la case vide où on a dessiné un petit disque, il n'est pas possible de poser 5,3,7 à cause de la ligne, et 2 à cause de la colonne, et 6 à cause de la région, il y a donc encore **4** possibilités (les valeurs 1,4,8,9)

*Travail attendu :*

0 - prendre en main le code fourni (fichier .h)

1 – Essayer de réaliser une grille de sudoku et écrivez l'algorithme que vous utilisez pour le faire.

2 - Vous devez coder ensuite les fonctions suivantes (voir le détail dans le fichier sudoku.h) :

| | | |
|---|---|---|
| uint8_t estValideH(sudoku s,uint8_t x,uint8_t y,uint8_t n); | Difficulté algorithmique : ★☆☆☆ | Difficulté technique : ☆☆☆☆ |
| uint8_t estValideV(sudoku s,uint8_t x,uint8_t y,uint8_t n); | Difficulté algorithmique : ★☆☆☆ | Difficulté technique : ☆☆☆☆ |
| uint8_t estValideR(sudoku s,uint8_t x,uint8_t y,uint8_t n); | Difficulté algorithmique : ★☆☆☆ | Difficulté technique : ☆☆☆☆ |
| uint8_t estValide(sudoku s,uint8_t x,uint8_t y,uint8_t n); | Difficulté algorithmique : ☆☆☆☆ | Difficulté technique : ☆☆☆☆ |
| uint8_t compterLibres(sudoku s); | Difficulté algorithmique : ★☆☆☆ | Difficulté technique : ☆☆☆☆ |
| void afficheSudoku(sudoku s); | Difficulté algorithmique : ☆☆☆☆ | Difficulté technique : ★☆☆☆ |
| uint8_t chargeSudoku(char path[]); | Difficulté algorithmique : ☆☆☆☆ | Difficulté technique : ★★☆☆ |
| void sauveSudoku(char path[],sudoku s); | Difficulté algorithmique : ☆☆☆☆ | Difficulté technique : ★★☆☆ |
| void initialiseSudoku(sudoku s); | Difficulté algorithmique : ☆☆☆☆ | Difficulté technique : ☆☆☆☆ |
| uint8_t lireAction(); | Difficulté algorithmique : ☆☆☆☆ | Difficulté technique : ★★☆☆ |

3 – Pour les plus rapides : Créer le fichier de ***tests*** et vérifier les résultats.