

# Day 7

30 November 2022 19:32

Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy **all of the following rules**:

Each of the digits **1-9** must occur exactly once in each row.

Each of the digits **1-9** must occur exactly once in each column.

Each of the digits **1-9** must occur exactly once in each of the 9 **3x3** sub-boxes of the grid.

The **'.'** character indicates empty cells.

Example 1:

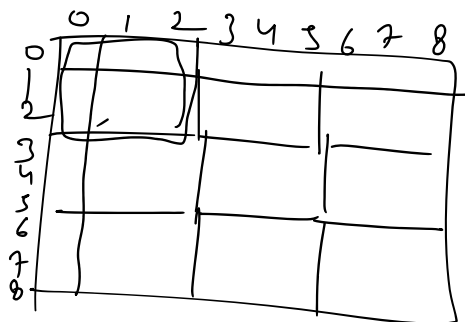
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

1-9

9x9

82 row 1-9 one time

82 column 1-9 one time



9 (3x3)

(0,6)	(0,7)	(0,8)
(1,6)	(1,7)	(1,8)
(2,6)	(2,7)	(2,8)

Handwritten notes and diagrams illustrating the solution process:

- Red wavy lines labeled 0, 1, 2, 3, 4, 5, 6, 7, 8.
- A red circle highlights the cell at row 0, column 6 (value 9) in the example grid.
- A red box highlights the 3x3 sub-grid starting at row 0, column 6.
- Red text: "row = 0", "col = 6".
- Red text: "row + 1".
- Red text: "col + 1".

1  
2

8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$\left[ 3 \times \left( \frac{\text{row}}{3} \right) + i \right]$$

$$\left[ 3 \times \left( \frac{\text{col}}{3} \right) + j \right]$$

(0,6)

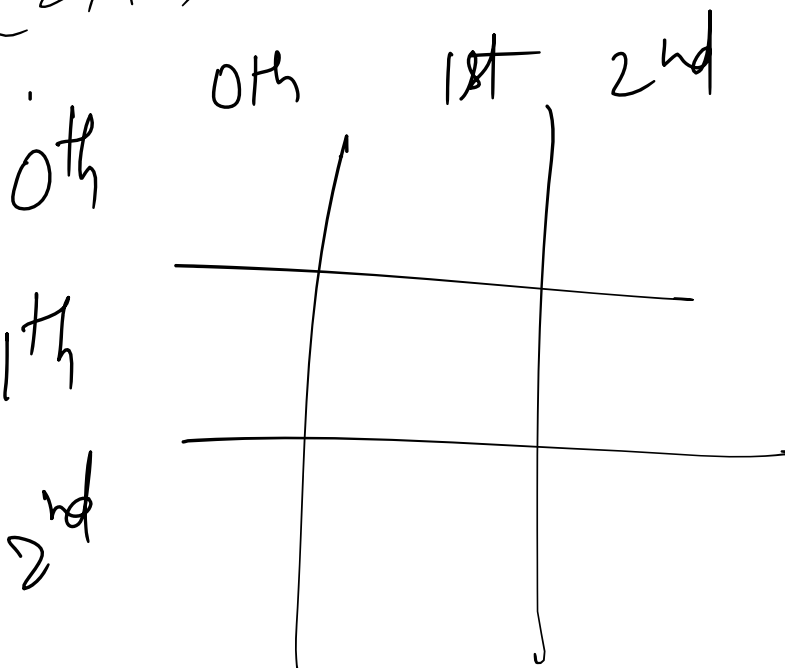
$$3 \times \left( \frac{0}{3} \right) \rightarrow 0^{\text{th}} \quad 0$$

$$3 \times \left( \frac{6}{3} \right) \rightarrow 2^{\text{nd}} \Rightarrow \underline{6}$$

$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$   $\begin{pmatrix} 0, 1 \end{pmatrix}$   $\begin{pmatrix} 0, 2 \end{pmatrix}$   
 $\begin{pmatrix} 1, 0 \end{pmatrix}$   $\begin{pmatrix} 1, 1 \end{pmatrix}$   $\begin{pmatrix} 1, 2 \end{pmatrix}$   
 $\begin{pmatrix} 2, 0 \end{pmatrix}$   $\begin{pmatrix} 2, 1 \end{pmatrix}$   $\begin{pmatrix} 2, 2 \end{pmatrix}$

, puzzle))

$\frac{d}{du}$



row \times 3 -

$$\left(\frac{0}{3}\right) \times 3 -$$

$$\left(\frac{6}{3}\right) \times 3$$

```

class Solution {
public:
    bool is_valid_position(int row, int col, char ch, vector<vector<char>> &board){
        // row
        for(int i = 0; i < 9; i++){
            if(board[row][i] == ch){
                return false;
            }
            if(board[i][col] == ch){
                return false;
            }
        }
        // gris 3*3
        for(int i = 0; i < 3; i++){
            for(int j = 0; j < 3; j++){
                if(board[i + (row/3)*3][j + (col/3)*3] == ch){
                    return false;
                }
            }
        }
        return true;
    }
    bool solver(vector<vector<char>> &board){
        for(int row = 0; row < 9; row++){
            for(int col = 0; col < 9; col++){
                if(board[row][col] == '.'){
                    for(int pos = 1; pos <= 9; pos++){
                        if(is_valid_position(row, col, '0'+pos, board)){
                            board[row][col] = '0' + pos;
                            if(solver(board)) return true;
                            board[row][col] = '.';
                        }
                    }
                    return false;
                }
            }
        }
        return true;
    }
    void solveSudoku(vector<vector<char>>& board) {
        solver(board);
    }
};

```