README

*This is a **SUDUKO SOLVER** . A standard Sudoku contains 81 cells, in a 9×9 grid, and has 9 boxes, each box being the intersection of the first, middle, or last 3 rows, and the first, middle, or last 3 columns. Each cell may contain a number from one to nine, and each number can only occur once in each row, column, and box. A Sudoku starts with some cells containing numbers (clues), and the goal is to solve the remaining cells. Proper Sudokus have one solution.*

*This project will first check whether the **SUDUKO** is correct or not (a number must be in and only in a row or column or in that particular sub-matrix). If that problem is valid ,then it'll give the particular solution of that problem , otherwise it will print "NO SOLUTION EXIST".*

TECHNIQUE USED:-

*For this project ,i've used "**BACKTRACKING**" technique.This algorithm visits the empty cells in some order, filling in digits sequentially, or backtracking when the number is found to be not valid.Briefly, a program would solve a puzzle by placing the digit "1" in the first cell and checking if it is allowed to be there. If there are no violations (checking row, column, and box constraints) then the algorithm advances to the next cell and places a "1" in that cell. When checking for violations, if it is discovered that the "1" is not allowed, the value is advanced to "2". If a cell is discovered where none of the 9 digits is allowed, then the algorithm leaves that cell blank and moves back to the previous cell. The value in that cell is then incremented by one. This is repeated until the allowed value in the last (81st) cell is discovered.*

CONSTRAINS:-

There are 3 constrains in suduko:-

*1)**ROW CONSTRAIN**:-Only one number can be in a row.*

*2)**COLUMN CONSTRAIN**:-Only one number can be in a column.*

*3)**BOX / SUB-MATRIX CONSTRAIN**:-Only one number can be in its particular submatrix.*

FUNCTIONS AND THERE WORK:-

main()-*This is driver function.This will take suduko as input and then pass it to valid_suduko(), If that suduko problem is valid then it'll pass it to sollveSuduko() and if it's solution exist then it'll pass it to printSuduko() and then print the soltion; otherwise print "no solution exist".*

valid_suduko()- *Function to check if the board invalid.*

SolveSudoku()- *Function to assign value to unassigned places and check for their validation.*

printSuduko()-*Function to print solution.*

valid_row()-*Function to check if a given row is valid. It will return: -1 if the row contains an invalid value, 0 if the row contains repeated valuesand 1 is the row is valid.*

valid_col()- *Function to check if a given column is valid. It will return:-1 if the column contains an invalid value ,0 if the column contains repeated values and 1 is the column is valid.*

valid_subsuduko()- *Function to check if all the subsquares are valid. It will return: -1 if a subsquare contains an invalid value 0 if a subsquare contains repeated values and 1 if the subsquares are valid.*

FindUnassignedLocation()- *Function finds an entry that is still* *unassigned in Suduko.If found,true is returned.If no unassigned entries remain,false is returned.*

isSafe()- *Checks whether it will be right to assign num to the given row and col.*

HOW CAN A USER RUN YOUR PROJECT AND USE IT:-

*If u want to use this program to solve suduko problems,you just have to run this code in your c++ IDE(vs code,codeblock,dev c++,etc).Then, you have to enter the suduko problem/question as an input using keyboard(exactly copy that problem as it is)(**use 0 for empty spaces**), and then press ENTER. After that you'll see whether that suduko problem is valid or not ;and if it is valid then you'll see the solution/answer of that problem on your display.*