## DAA LAB

## LAB 9: Implementation of any scheme to solve the ***SUDOKU*** puzzle.

**PROGRAM:**

**#include<iostream>**

**#define N 9**

**using namespace std;**

**int grid[N][N] = {**

**{3, 0, 6, 5, 0, 8, 4, 0, 0},**

**{5, 2, 0, 0, 0, 0, 0, 0, 0},**

**{0, 8, 7, 0, 0, 0, 0, 3, 1},**

**{0, 0, 3, 0, 1, 0, 0, 8, 0},**

**{9, 0, 0, 8, 6, 3, 0, 0, 5},**

**{0, 5, 0, 0, 9, 0, 6, 0, 0},**

**{1, 3, 0, 0, 0, 0, 2, 5, 0},**

**{0, 0, 0, 0, 0, 0, 0, 7, 4},**

**{0, 0, 5, 2, 0, 6, 3, 0, 0}**

**};**

**bool isPresentInCol(int col, int num)**

**{**

**//check whether num is present in col or not**

**for (int row = 0; row < N; row++)**

**if (grid[row][col] == num)**

**return true;**

**return false;**

**}**

**bool isPresentInRow(int row, int num)**

**{**

**//check whether num is present in row or not**

**for (int col = 0; col < N; col++)**

**if (grid[row][col] == num)**

**return true;**

**return false;**

**}**

**bool isPresentInBox(int boxStartRow, int boxStartCol, int num)**

**{**

**//check whether num is present in 3x3 box or not**

**for (int row = 0; row < 3; row++)**

**for (int col = 0; col < 3; col++)**

**if (grid[row+boxStartRow][col+boxStartCol] == num)**

**return true;**

**return false;**

**}**

**void sudokuGrid()**

**{**

**//print the sudoku grid after solve**

**for (int row = 0; row < N; row++){**

**for (int col = 0; col < N; col++){**

**if(col == 3 || col == 6)**

**cout << " | ";**

**cout << grid[row][col] <<" ";**

**}**

**if(row == 2 || row == 5){**

**cout << endl;**

**for(int i = 0; i<N; i++)**

**cout << "---";**

**}**

**cout << endl;**

**}**

**}**

**bool findEmptyPlace(int &row, int &col)**

**{**

**//get empty location and update row and column**

**for (row = 0; row < N; row++)**

**for (col = 0; col < N; col++)**

**if (grid[row][col] == 0) //marked with 0 is empty**

**return true;**

**return false;**

**}**

**bool isValidPlace(int row, int col, int num)**

**{**

**//when item not found in col, row and current 3x3 box**

**return !isPresentInRow(row, num) && !isPresentInCol(col, num) && !isPresentInBox(row - row%3 ,**

**col - col%3, num);**

**}**

**bool solveSudoku()**

**{**

**int row, col;**

**if (!findEmptyPlace(row, col))**

**return true;**

**//when all places are filled**

**for (int num = 1; num <= 9; num++)**

**{**

**//valid numbers are 1 - 9**

**if (isValidPlace(row, col, num))**

**{**

**//check validation, if yes, put the number in the grid**

**grid[row][col] = num;**

**if (solveSudoku())**

**//recursively go for other rooms in the grid**

**return true;**

**grid[row][col] = 0;**

**//turn to unassigned space when conditions are not satisfied**

**}**

**}**

**return false;**

**}**

**int main(){**

**if (solveSudoku() == true)**

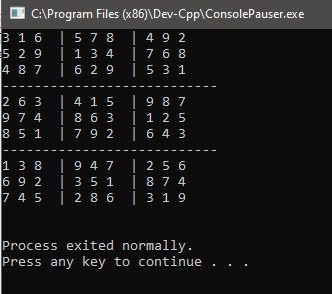
**sudokuGrid();**

**else**

**cout << "No solution exists";**

**}**

**OUTPUT:**

****