SUDOKU PROJECT

#include <iostream>

#include <vector>

#define N 9  // Size of the Sudoku grid (9x9)

using namespace std;

// Function to print the Sudoku grid

void printGrid(const vector<vector<int>>& grid) {

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

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

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

        }

        cout << endl;

    }

}

// Function to check if it's safe to place a number in a cell

bool isSafe(const vector<vector<int>>& grid, int row, int col, int num) {

    // Check if the number is not in the current row or column

    for (int x = 0; x < N; x++) {

        if (grid[row][x] == num || grid[x][col] == num) {

            return false;

        }

    }

    // Check if the number is not in the current 3x3 subgrid

    int startRow = row - row % 3, startCol = col - col % 3;

    for (int i = 0; i < 3; i++) {

        for (int j = 0; j < 3; j++) {

            if (grid[i + startRow][j + startCol] == num) {

                return false;

            }

        }

    }

    return true;

}

// Function to solve the Sudoku puzzle using backtracking

bool solveSudoku(vector<vector<int>>& grid, int row, int col) {

    // If we have reached the last cell, the Sudoku is solved

    if (row == N - 1 && col == N) {

        return true;

    }

    // Move to the next row if we've reached the end of the current row

    if (col == N) {

        row++;

        col = 0;

    }

    // Skip cells that are already filled

    if (grid[row][col] != 0) {

        return solveSudoku(grid, row, col + 1);

    }

    // Try placing numbers 1-9 in the current empty cell

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

        if (isSafe(grid, row, col, num)) {

            grid[row][col] = num;

            // Recursively try to solve the rest of the grid

            if (solveSudoku(grid, row, col + 1)) {

                return true;

            }

            // Backtrack if placing num doesn't lead to a solution

            grid[row][col] = 0;

        }

    }

    return false;  // Trigger backtracking

}

int main() {

    // Example Sudoku grid with 0s representing empty cells

    vector<vector<int>> grid = {

        {5, 3, 0, 0, 7, 0, 0, 0, 0},

        {6, 0, 0, 1, 9, 5, 0, 0, 0},

        {0, 9, 8, 0, 0, 0, 0, 6, 0},

        {8, 0, 0, 0, 6, 0, 0, 0, 3},

        {4, 0, 0, 8, 0, 3, 0, 0, 1},

        {7, 0, 0, 0, 2, 0, 0, 0, 6},

        {0, 6, 0, 0, 0, 0, 2, 8, 0},

        {0, 0, 0, 4, 1, 9, 0, 0, 5},

        {0, 0, 0, 0, 8, 0, 0, 7, 9}

    };

    // Attempt to solve the Sudoku

    if (solveSudoku(grid, 0, 0)) {

        printGrid(grid);

    } else {

        cout << "No solution exists." << endl;

    }

}