**3.Sudoku Solver**

**Write a program that solves Sudoku puzzles. Implement an algorithm to solve puzzles of varying difficulty levels. You can also create a graphical user interface (GUI) for user interaction**.

**CODE**

#include <iostream>

using namespace std;

#define N 9 // Grid size

// Function to print the Sudoku grid

void printGrid(int grid[N][N])

{

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

bool isSafe(int grid[N][N], int row, int col, int num)

{

for (int x = 0; x < N; x++)

{

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

return false;

}

// Check 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;

}

// Backtracking solver

bool solveSudoku(int grid[N][N])

{

int row, col;

bool emptySpot = false;

// Find an empty cell

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

{

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

{

if (grid[row][col] == 0)

{

emptySpot = true;

break;

}

}

if (emptySpot) break;

}

if (!emptySpot) return true;

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

{

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

{

grid[row][col] = num;

if (solveSudoku(grid))

return true;

grid[row][col] = 0; // Backtrack

}

}

return false;

}

int main() {

// Easy Sudoku

int easy[N][N] = {

{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}

};

// Medium Sudoku

int medium[N][N] = {

{0, 0, 0, 2, 0, 0, 0, 6, 3},

{3, 0, 0, 0, 0, 5, 4, 0, 1},

{0, 0, 1, 0, 0, 3, 9, 8, 0},

{0, 0, 0, 0, 0, 0, 0, 9, 0},

{0, 0, 0, 5, 3, 8, 0, 0, 0},

{0, 3, 0, 0, 0, 0, 0, 0, 0},

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

{5, 0, 3, 7, 0, 0, 0, 0, 8},

{4, 7, 0, 0, 0, 1, 0, 0, 0}

};

// Hard Sudoku

int hard[N][N] = {

{0, 0, 0, 6, 0, 0, 4, 0, 0},

{7, 0, 0, 0, 0, 3, 6, 0, 0},

{0, 0, 0, 0, 9, 1, 0, 8, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 5, 0, 1, 8, 0, 0, 0, 3},

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

{0, 4, 0, 2, 0, 0, 0, 6, 0},

{9, 0, 3, 0, 0, 0, 0, 0, 0},

{0, 2, 0, 0, 0, 0, 1, 0, 0}

};

int choice;

cout << "Select Difficulty Level:\n";

cout << "1. Easy\n2. Medium\n3. Hard\n";

cout << "Enter choice (1-3): ";

cin >> choice;

int (\*selectedGrid)[N];

switch (choice)

{

case 1:

selectedGrid = easy;

cout << "\nSolving Easy Sudoku...\n";

break;

case 2:

selectedGrid = medium;

cout << "\nSolving Medium Sudoku...\n";

break;

case 3:

selectedGrid = hard;

cout << "\nSolving Hard Sudoku...\n";

break;

default:

cout << "Invalid choice. Exiting.\n";

return 0;

}

cout << "Original Sudoku Grid:\n";

printGrid(selectedGrid);

if (solveSudoku(selectedGrid))

{

cout << "\nSolved Sudoku Grid:\n";

printGrid(selectedGrid);

} else

{

cout << "No solution exists.\n";

}

return 0;

}

**OUTPUT**





