Mangalore institute of technology and engineering , moodabidri

Group-cp044

Group members:

Sparshitha

Shanthika

Yamuna

shravya

Shreya

Vikas

Problem statement

Sudoku solver

Implement a function to solve Sudoku puzzles programmatically apply Sudoku solver algorithm and display the solve Sudoku grid to the user

Introduction to Sudoku.

Sudoku is a puzzle that has enjoyed worldwide popularity since 2005. To solve a Sudoku puzzle, one needs to use a combination of logic and trial-and-error. More maths is involved behind the scenes: combinatorics used in counting valid Sudoku grids, group theory used to describe ideas of when two grids are equivalent, and computational complexity with regards to solving Sudoku.

Although sudoku-type patterns had been used earlier in agricultural design, their first appearance in puzzle form was in 1979 in a New York-based puzzle magazine, which called them Number Place puzzles. They next appeared in 1984 in a magazine in Japan, where they acquire the name Sudoku (abbreviated from *suuji wa dokushin ni kagiru*, meaning “the numbers must remain single”). In spite of the puzzle’s popularity in Japan, the worldwide sudoku explosion had to wait another 20 years.

IDEA OF SUDOKU:

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

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

WORK DISTRIBUTION:

> MINDMAP:

1.SHRAVYA

2.SHREYA

> CODE CREATION:

1.SPARSHITHA

2.SHANTHIKA

3.YAMUNA

> REPORT:

1.SPARSHITHA

2.SHANTHIKA

3.YAMUNA

4.SHRAVYA

5.SHREYA

6.VIKAS

ABOUT OUR PROJECT:

Our project is typically is a Sudoku game .A game which you all are familiar with, Here we have a user and admin. Sudoku game is all about filling the numbers with some certain rules and technique.Some permissions are only for the the admin and some are unique for the users.

The permissions which **Users** have is to:

Login

To enter the order of puzzles which they have to play

To set the time they wish complete their puzzle in

Change the order of difficulty

To delete the puzzle or to enter the numbers

The permissions which **Admin** have is to:

Login

Maintain ,monitoring and moderation

Bux fixes and updates

Game configuration

Custom and support

Security

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

// Constants
#define GRID_SIZE 9
#define EMPTY_CELL 0

// Function prototypes
bool solveSudoku(int grid[GRID_SIZE][GRID_SIZE]);
bool isValidMove(int grid[GRID_SIZE][GRID_SIZE], int row, int col, int num);
void printGrid(int grid[GRID_SIZE][GRID_SIZE]);
bool loginUser();
bool isAdminLoggedIn = false;

int main() {
    int sudokuGrid[GRID_SIZE][GRID_SIZE] = {
        {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}
    };

    int choice;
    printf("Welcome to Sudoku Game\n");
    do {
        printf("Main Menu\n");
        printf("1. Login\n");
        printf("2. Play Sudoku\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                if (loginUser()) {
                    printf("Login successful.\n");
                } else {
                    printf("Login failed.\n");
```

```
            }
            break;
        case 2:
            if (isAdminLoggedIn) {
                // Play Sudoku
                printf("Sudoku Puzzle:\n");
                printGrid(sudokuGrid);
                printf("Solving Sudoku...\n");
                if (solveSudoku(sudokuGrid)) {
                    printf("Solved Sudoku:\n");
                    printGrid(sudokuGrid);
                } else {
                    printf("No solution found.\n");
                }
            } else {
                printf("You must log in first.\n");
            }
            break;
        case 3:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
            break;
    }
  } while (choice != 3);

  return 0;
}

// Sudoku Solver Functions

bool solveSudoku(int grid[GRID_SIZE][GRID_SIZE]) {
  int row, col;

  // Find an empty cell
  bool isEmpty = false;
  for (row = 0; row < GRID_SIZE; row++) {
    for (col = 0; col < GRID_SIZE; col++) {
      if (grid[row][col] == EMPTY_CELL) {
        isEmpty = true;
        break;
      }
    }
    if (isEmpty)
      break;
```

```
    }

    // If no empty cell is found, the puzzle is solved
    if (!isEmpty)
        return true;

    // Try placing numbers 1-9 in the empty cell
    for (int num = 1; num <= 9; num++) {
        if (isValidMove(grid, row, col, num)) {
            grid[row][col] = num;

            if (solveSudoku(grid))
                return true;

            // If the current configuration does not lead to a solution, backtrack
            grid[row][col] = EMPTY_CELL;
        }
    }

    return false;
}

bool isValidMove(int grid[GRID_SIZE][GRID_SIZE], int row, int col, int num) {
    // Check if 'num' is not already present in the current row, column, and 3x3 grid
    for (int i = 0; i < GRID_SIZE; i++) {
        if (grid[row][i] == num || grid[i][col] == num || grid[row - row % 3 + i / 3][col - col % 3 + i
% 3] == num) {
            return false;
        }
    }
    return true;
}

void printGrid(int grid[GRID_SIZE][GRID_SIZE]) {
    for (int i = 0; i < GRID_SIZE; i++) {
        for (int j = 0; j < GRID_SIZE; j++) {
            printf("%2d ", grid[i][j]);
        }
        printf("\n");
    }
}

// Login Function

bool loginUser() {
    char username[20];
```

```
    char password[20];

    printf("Enter username: ");
    scanf("%s", username);
    printf("Enter password: ");
    scanf("%s", password);

    // Simulated username and password check (Replace with secure authentication)
    if (strcmp(username, "user") == 0 && strcmp(password, "password") == 0) {
        isAdminLoggedIn = true;
        return true;
    } else {
        isAdminLoggedIn = false;
        return false;
    }
} #include <stdio.h>
#include <stdbool.h>
#include <string.h>

// Constants
#define GRID_SIZE 9
#define EMPTY_CELL 0

// Function prototypes
bool solveSudoku(int grid[GRID_SIZE][GRID_SIZE]);
bool isValidMove(int grid[GRID_SIZE][GRID_SIZE], int row, int col, int num);
void printGrid(int grid[GRID_SIZE][GRID_SIZE]);
bool loginUser();
bool isAdminLoggedIn = false;

int main() {
    int sudokuGrid[GRID_SIZE][GRID_SIZE] = {
        {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}
    };

    int choice;
    printf("Welcome to Sudoku Game\n");
    do {
```

```
        printf("Main Menu\n");
        printf("1. Login\n");
        printf("2. Play Sudoku\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                if (loginUser()) {
                    printf("Login successful.\n");
                } else {
                    printf("Login failed.\n");
                }
                break;
            case 2:
                if (isAdminLoggedIn) {
                    // Play Sudoku
                    printf("Sudoku Puzzle:\n");
                    printGrid(sudokuGrid);
                    printf("Solving Sudoku...\n");
                    if (solveSudoku(sudokuGrid)) {
                        printf("Solved Sudoku:\n");
                        printGrid(sudokuGrid);
                    } else {
                        printf("No solution found.\n");
                    }
                } else {
                    printf("You must log in first.\n");
                }
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    } while (choice != 3);

    return 0;
}

// Sudoku Solver Functions

bool solveSudoku(int grid[GRID_SIZE][GRID_SIZE]) {
    int row, col;
```

```
    // Find an empty cell
    bool isEmpty = false;
    for (row = 0; row < GRID_SIZE; row++) {
        for (col = 0; col < GRID_SIZE; col++) {
            if (grid[row][col] == EMPTY_CELL) {
                isEmpty = true;
                break;
            }
        }
        if (isEmpty)
            break;
    }

    // If no empty cell is found, the puzzle is solved
    if (!isEmpty)
        return true;

    // Try placing numbers 1-9 in the empty cell
    for (int num = 1; num <= 9; num++) {
        if (isValidMove(grid, row, col, num)) {
            grid[row][col] = num;

            if (solveSudoku(grid))
                return true;

            // If the current configuration does not lead to a solution, backtrack
            grid[row][col] = EMPTY_CELL;
        }
    }

    return false;
}

bool isValidMove(int grid[GRID_SIZE][GRID_SIZE], int row, int col, int num) {
    // Check if 'num' is not already present in the current row, column, and 3x3 grid
    for (int i = 0; i < GRID_SIZE; i++) {
        if (grid[row][i] == num || grid[i][col] == num || grid[row - row % 3 + i / 3][col - col % 3 + i
% 3] == num) {
            return false;
        }
    }
    return true;
}

void printGrid(int grid[GRID_SIZE][GRID_SIZE]) {
```

```
    for (int i = 0; i < GRID_SIZE; i++) {
        for (int j = 0; j < GRID_SIZE; j++) {
            printf("%2d ", grid[i][j]);
        }
        printf("\n");
    }
}

// Login Function

bool loginUser() {
    char username[20];
    char password[20];

    printf("Enter username: ");
    scanf("%s", username);
    printf("Enter password: ");
    scanf("%s", password);

    // Simulated username and password check (Replace with secure authentication)
    if (strcmp(username, "user") == 0 && strcmp(password, "password") == 0) {
        isAdminLoggedIn = true;
        return true;
    } else {
        isAdminLoggedIn = false;
        return false;
    }
}
```