VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590 018

A Project Work Report
On

"SUDUKO GAMING"

By

SHRAVYA	4MT21CS150
SHREYA	4MT21CS152
SHANTIKA	4MT21CS139
SPARSHITHA	4MT21CS161
YAMUNA	4MT21CS188
VIKAS	4MT21CS180

MITE
Invent Solutions

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING
Badaga Mijar, Moodabidri-574 225, Karnataka

2022-2023

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution

(A Unit of Rajalaxmi Education Trust)

Affiliated to Visvesvaraya Technological University, Belagavi Badaga

Mijar, Moodabidri – 574 225 , Karnataka

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We,SHRAVYA(4MT21CS150),SHREYA(4MT21CS152),SHANTIKA(4MT21CS139),SPARSHITHA(4MT21CS161),YAMUNA(4MT21CS188),VIKAS(4MT21CS180),students of 4th semester BE in Computer Science & Engineering, **Mangalore Institute of Technology and Engineering, Moodabidri,** here by declare that the project work entitled "**SUDUKO GAMING",** submitted to the department of Computer Science & Engineering during the academic year **2022-23**. This project work is submitted as assignment to **Design and Analysis of Algorithms(21CS42).**

Date14/09/2023

PLACE : MANGALORE INSTITUTE OF TECHNOLOGY ,MOODABIDRI

SPARSHITHA
SHREYA
SHANTIKA
SHRAVYA
YAMUNA
VIKAS

TABLE OF CONTENTS

TITLE:

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

ABSTRACT:

Sudoku, a popular logic-based number puzzle, has captivated the minds of puzzle enthusiasts and mathematicians alike. This paper delves into the world of Sudoku, exploring its rich history, mathematical properties, and solving strategies. We begin by tracing the origins of Sudoku and its evolution into the modern puzzle we know today.

Our research focuses on dissecting the underlying patterns and rules that govern Sudoku puzzles, shedding light on the algorithmic complexity and computational aspects of solving them. We discuss various solving strategies, ranging from basic techniques to advanced methods employed by experts. Through a detailed analysis of puzzle generation algorithms, we investigate the unique characteristics that make Sudoku puzzles challenging and engaging.

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.

2.2 Objectives

Sudoku is a popular logic-based number-placement puzzle that is played on a 9x9 grid divided into nine 3x3 subgrids or regions. The primary objectives of Sudoku are:

1. Fill the Grid: The main objective of Sudoku is to fill the entire 9x9 grid with numbers so that each row, each column, and each of the nine 3x3 subgrids (also called boxes or regions) contains all of the digits from 1 to 9. In other words, no digit can be repeated in any row, column, or 3x3 subgrid.

2. Logic and Deduction: Sudoku is a puzzle that requires logical deduction and problem-solving skills to solve.

3. No Guessing: A fundamental rule in Sudoku is that guessing is not allowed. Players must use logic and reasoning to determine the placement of each number

4. Solvable with Pure Logic: A well-constructed Sudoku puzzle can be solved using pure logical reasoning, without the need for random guessing or trial-and-error.

5. A Single Solution: Every valid Sudoku puzzle has a unique solution.

6. Difficulty Levels: Sudoku puzzles come in various difficulty levels, ranging from easy to very challenging. The objectives remain the same regardless of the difficulty level, but the complexity of logical reasoning required increases as the puzzles get harder.

3. Technologies Used

- Programming Language: C

- Standard Libraries: `<stdio.h>`, `<stdlib.h>`, `<string.h>`

Digital Platforms: Sudoku is often played on digital platforms, including websites, mobile apps, and computer software. .

4. System

Architecture 4.1

Front-End

The front-end of this project is text-based and relies on user input and output through the command line.

4.2 Back-End

The back-end manages user authentication, menu navigation, and element manipulation.

4.3 Database

The system does not use a database as it stores user and element data in memory.

5. Project Modules

1.Sudoku Board Representation: Define a data structure to represent the Sudoku board. You can use a 2D array to represent the grid. Each cell can hold a number (1-9) or a placeholder for an empty cell.

2. Board Initialization:

Create a module to initialize the Sudoku board. You can start with an empty board or generate a random valid Sudoku puzzle to solver

3. Input and Output:

Implement functions to read and display the Sudoku board. This could include reading from a file or taking user input.

4. Sudoku Solver: Implement a Sudoku solver using backtracking or some other algorithm. The solver should be able to solve any valid Sudoku puzzle.

5. User Interface: Create a simple text-based or graphical user interface for the user to interact with the Sudoku puzzle. Allow them to input their own puzzles or request solutions.

6. Sudoku Generator: Develop a Sudoku puzzle generator that can create puzzles of varying difficulty levels. This involves starting with a solved Sudoku grid and then removing numbers to create the puzzle.

7. Difficulty Levels: Implement a module to classify the difficulty level of a Sudoku puzzle based on the solving techniques required.

8. Validation: Include a module to validate whether the current state of the Sudoku board is a valid solution or not. This includes checking rows, columns, and 3x3 subgrids for duplicates.

9. Error Handling:Implement error handling to catch invalid user inputs and Sudoku board states.

6. Design and Implementation

6.1 Front-End Design

- The front-end design is minimalistic and relies on text-based input and output.

6.2 Back-End Design

- The back-end handles user login, menu selection, and element manipulation.

6.3 Database Design

- The system does not use a database.

7. Features and Functionality

7.1 Feature 1: User Login

- Users can log in with a username and password.

7.2 Feature 2: Menu Navigation

- Administrators and customers have separate menus.

8. Testing

8.1 Unit Testing

- Each function has been tested individually.

8.2 Integration Testing

- The integration of functions has been tested.

8.3 User Acceptance Testing

- The system has been tested with user input to ensure it functions as expected.

9. Challenges Faced

- Challenges included handling user input and implementing search algorithms.

In conclusion:

Sudoku is a challenging and engaging puzzle game that has captivated the minds of millions worldwide. It's a game of logic, patience, and problem-solving, where players must fill a grid with numbers while adhering to strict rules. Sudoku offers a mental workout that can sharpen cognitive skills such as deductive reasoning and pattern recognition. Sudoku remains a beloved puzzle game that has stood the test of time, offering a unique blend of entertainment, mental stimulation, and educational value. Whether played casually or pursued as a serious endeavor, Sudoku continues to captivate minds and bring joy to countless individuals around the world.

REFERENCES:

You can find various tutorials, articles, and example code online to help you implement a Sudoku solver in C. Here are some resources:

- Sudoku Solver Tutorial by Norvig: Link (Python-based, but the logic can be adapted to C)
- GitHub Repositories with C Sudoku Solvers: Search for open-source Sudoku solver implementations on GitHub.
- C Programming Books: Consider referring to C programming books for algorithmic and data structure concepts.

APPENDICES:

Appendices are often used to provide supplementary information or resources related to a document or research paper. When it comes to Sudoku, appendices might include additional puzzles, solution strategies, or any other relevant information that enhances the reader's understanding of Sudoku. Here are some ideas for appendices on Sudoku:

Sudoku Puzzle Variations: Include different types of Sudoku puzzles like Irregular Sudoku, Killer Sudoku, Samurai Sudoku, and Diagonal Sudoku. Provide examples and rules for each variation.

Sudoku Solving Techniques: Explain various strategies and techniques for solving Sudoku puzzles. This could include easy-to-understand step-by-step guides for beginners and more advanced techniques for seasoned Sudoku players.
Sudoku Software and Apps: List and describe popular Sudoku-solving software and mobile apps. Include information on where to download or purchase them.

Sudoku History: Give a brief history of Sudoku, its origins, and how it became a global phenomenon.

Sudoku Competitions and Events: Provide information about Sudoku tournaments, competitions, and events, including details about how to participate or watch.

Sudoku Books and References: Suggest books, websites, and other references for those interested in learning more about Sudoku, improving their skills, or exploring advanced techniques.

Sudoku Templates: Offer printable Sudoku grid templates for readers who want to create their puzzles or practice without any predefined numbers.

Sudoku Glossary: Include a glossary of terms commonly used in Sudoku puzzles, such as "naked pairs," "hidden triples," "box-line reduction," and others, along with clear explanations.

Sudoku Tips and Tricks: Share practical tips and tricks for solving Sudoku puzzles efficiently, such as how to recognize common patterns and avoid common mistakes.

Sudoku Resources: Provide a list of online resources, forums, and communities where Sudoku enthusiasts can discuss strategies, share puzzles, and seek help.

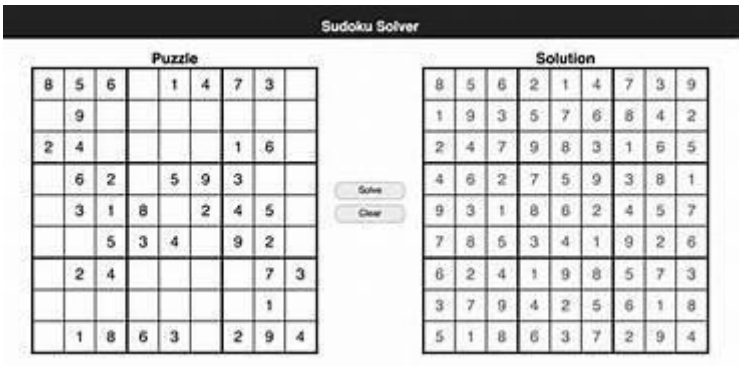
10. Future Enhancements

- Future enhancements could include data persistence with file storage and a more user-friendly interface.

IDEA OF SUDOKU:

1. Grid Structure: A standard Sudoku grid consists of 81 cells arranged in a 9x9 grid. This grid is divided into nine 3x3 regions, each containing nine cells. These regions are typically shaded or outlined to distinguish them from the rest of the grid.
2. Initial Numbers: A Sudoku puzzle begins with some of the cells already filled with numbers. These numbers are called "given numbers" or "clues." The placement of these initial numbers is such that the puzzle has a unique solution.
3. Number Placement: To solve a Sudoku puzzle, you must fill in the empty cells with numbers from 1 to 9. The placement of numbers must adhere to the following rules:
    - Each row must contain all the numbers from 1 to 9, with no repetitions.
4. Solving Techniques: Sudoku puzzles can vary in difficulty. Solving techniques range from simple logical deductions to more advanced strategies. Some common solving techniques include:
    - Scanning: Identifying the only possible candidate for a cell based on the numbers already present in the row, column, and region.
    - Crosshatching: Focusing on rows and columns with fewer missing numbers to narrow down possibilities. .

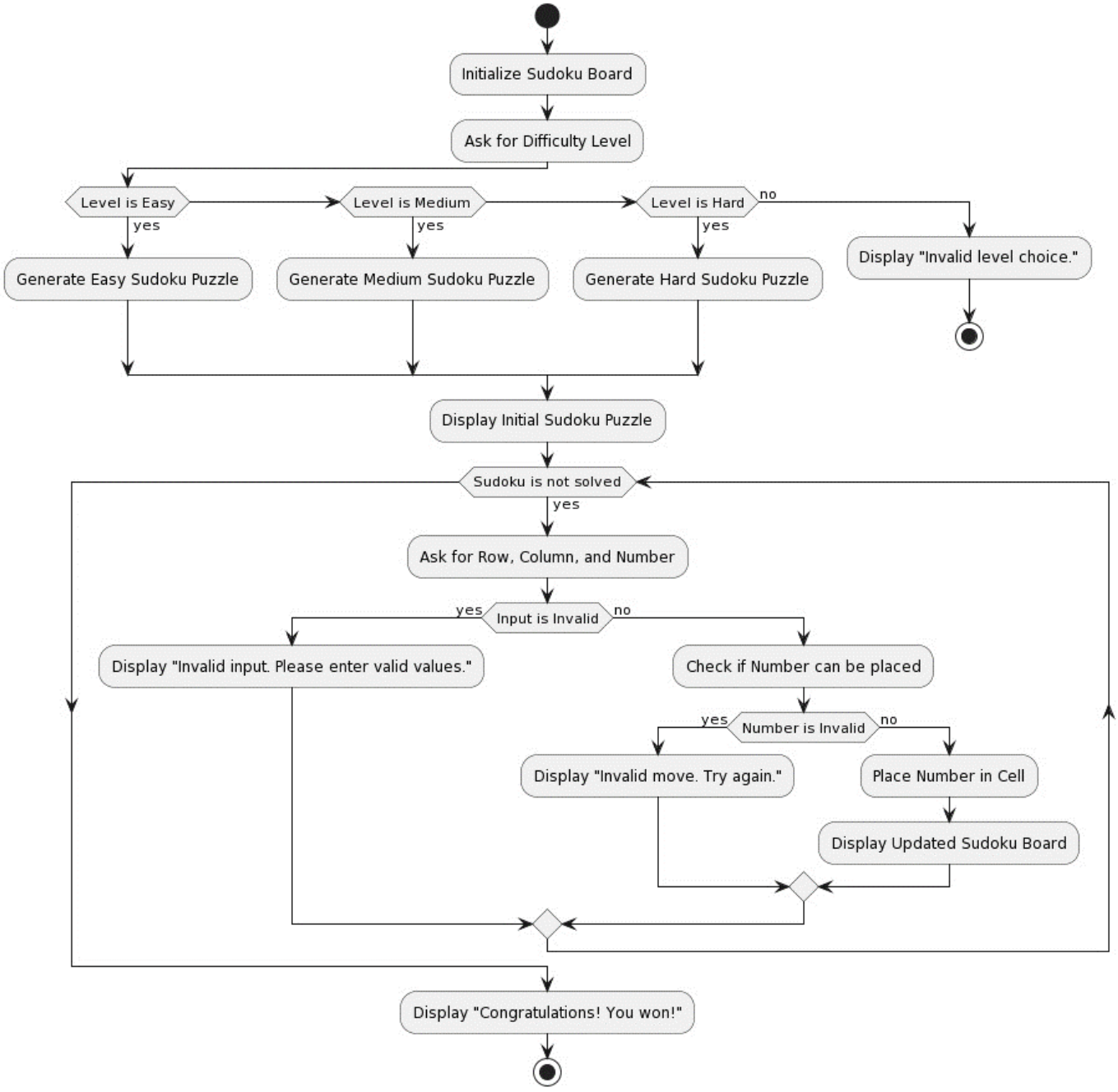
5. Solvability: A well-constructed Sudoku puzzle should have a unique solution that can be reached through logic and deduction alone. Some puzzles are designed to be easier, while others are more challenging and require more advanced solving techniques.



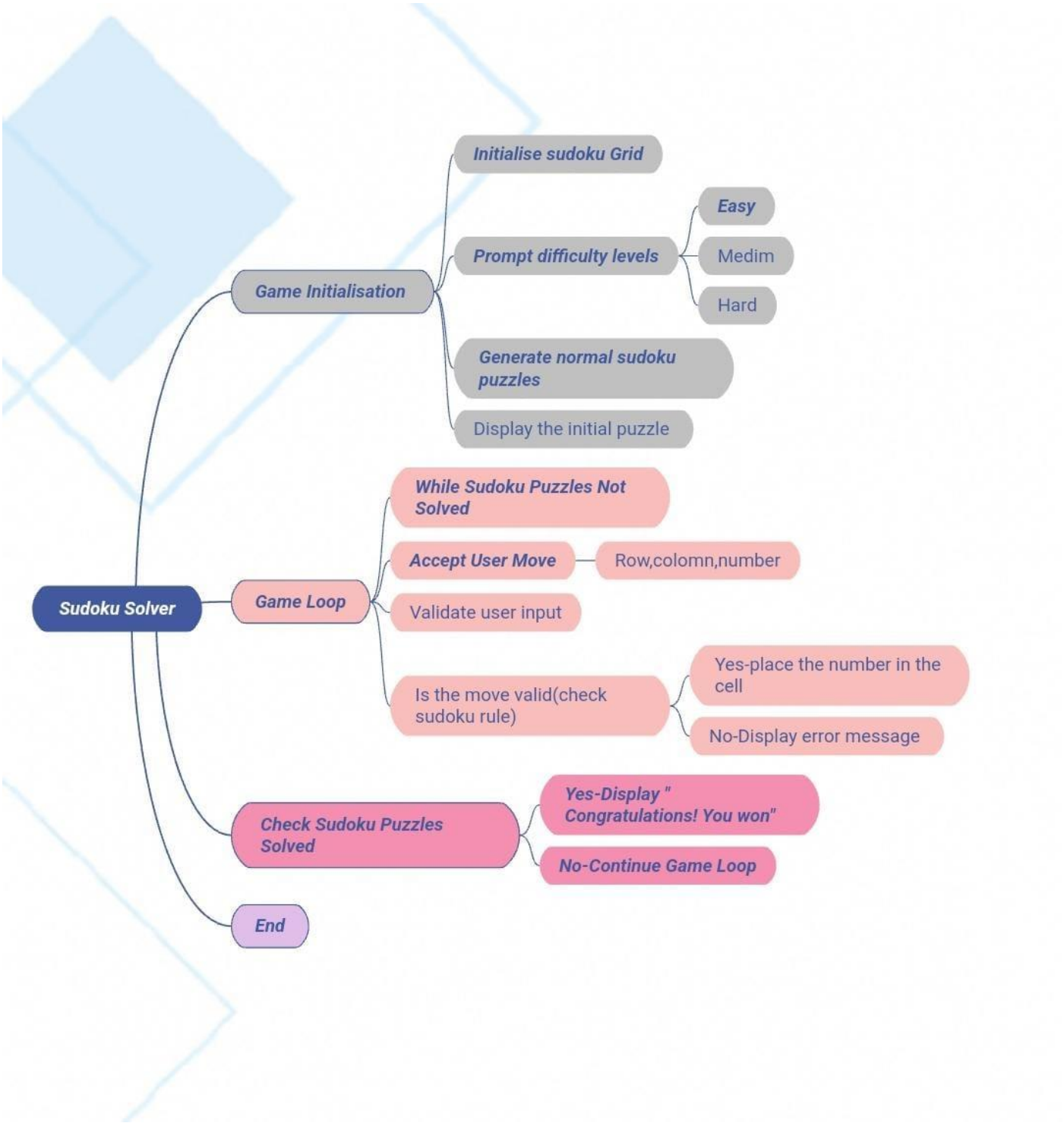
ABOUT OUR PROJECT:

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.

FLOWCHART:


Initialize Sudoku Board
Ask for Difficulty Level
Level is Easy
yes
Level is Medium
yes
Level is Hard
yes
no
Display "Invalid level choice."
Generate Easy Sudoku Puzzle
Generate Medium Sudoku Puzzle
Generate Hard Sudoku Puzzle
Display Initial Sudoku Puzzle
Sudoku is not solved
yes
Ask for Row, Column, and Number
yes
Input is Invalid
no
Display "Invalid input. Please enter valid values."
Check if Number can be placed
yes
Number is Invalid
no
Display "Invalid move. Try again."
Place Number in Cell
Display Updated Sudoku Board
Display "Congratulations! You won!"

MINDMAP:


Sudoku Solver
Game Initialisation
Initialise sudoku Grid
Prompt difficulty levels
Easy
Medim
Hard
Generate normal sudoku puzzles
Display the initial puzzle
Game Loop
While Sudoku Puzzles Not Solved
Accept User Move
Row,colomn,number
Validate user input
Is the move valid(check sudoku rule)
Yes-place the number in the cell
No-Display error message
Check Sudoku Puzzles Solved
Yes-Display " Congratulations! You won"
No-Continue Game Loop
End

CODE SNIPPETS:

```
#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>


// Define the size of the Sudoku grid

#define N 9


// Function to print the Sudoku board

void printBoard(int board[N][N]) {

    printf("\nSudoku Board:\n");

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

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

            printf("%2d ", board[i][j]);

        }

        printf("\n");

    }

}


// Function to check if a number can be placed in a given cell

bool isValid(int board[N][N], int row, int col, int num) {

    // Check the row and column
```

```
    for (int i = 0; i < N; i++) {
        if (board[row][i] == num || board[i][col] == num) {
            return false;
        }
    }

    // Check the 3x3 grid
    int startRow = (row / 3) * 3;
    int startCol = (col / 3) * 3;
    for (int i = startRow; i < startRow + 3; i++) {
        for (int j = startCol; j < startCol + 3; j++) {
            if (board[i][j] == num) {
                return false;
            }
        }
    }

    return true;
}

// Function to generate a randomized Sudoku puzzle
void generateRandomSudoku(int board[N][N], int level) {
    // Clear the board
```

```
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            board[i][j] = 0;
        }
    }

    // Seed the random number generator
    srand(time(NULL));

    // Generate a Sudoku puzzle by filling random cells
    int filledCells = 0;
    while (filledCells < level) {
        int row = rand() % N;
        int col = rand() % N;
        int num = rand() % 9 + 1;

        if (isValid(board, row, col, num) && board[row][col] == 0) {
            board[row][col] = num;
            filledCells++;
        }
    }
}
```

```
// Function to check if the Sudoku puzzle is solved
bool isSolved(int board[N][N]) {
    // Check if all cells are filled
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (board[i][j] == 0) {
                return false;
            }
        }
    }

    // Check if the filled board is valid
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            int num = board[i][j];
            board[i][j] = 0; // Temporarily remove the number
            if (!isValid(board, i, j, num)) {
                board[i][j] = num; // Restore the number
                return false;
            }
            board[i][j] = num; // Restore the number
        }
    }
```

```
    return true;
}

int main() {
    int board[N][N];
    int level;

    printf("Welcome to Sudoku!\n");
    printf("Select the difficulty level (1: Easy, 2: Medium, 3: Hard): ");
    scanf("%d", &level);

    switch (level) {
        case 1:
            generateRandomSudoku(board, 30); // Easy level
            break;
        case 2:
            generateRandomSudoku(board, 40); // Medium level
            break;
        case 3:
            generateRandomSudoku(board, 50); // Hard level
            break;
        default:
```

```
        printf("Invalid level choice.\n");
        return 1;
}

// Display the initial Sudoku puzzle
printBoard(board);

// Main game loop
while (!isSolved(board)) {
    int row, col, num;
    printf("Enter row (1-9), column (1-9), and number (1-9) separated by spaces (e.g., 3 5 7): ");
    scanf("%d %d %d", &row, &col, &num);

    // Check if the input is valid
    if (row < 1 || row > 9 || col < 1 || col > 9 || num < 1 || num > 9) {
        printf("Invalid input. Please enter valid values.\n");
        continue;
    }

    // Convert to 0-based indexing
    row--;
    col--;
```

```
        // Check if the number can be placed in the selected cell
        if (!isValid(board, row, col, num)) {
            printf("Invalid move. Try again.\n");
            continue;
        }

        // Place the number in the cell
        board[row][col] = num;

        // Display the updated Sudoku board
        printBoard(board);
    }

    printf("Congratulations! You won!\n");

    return 0;
}
```

SCREENSHOTS: