JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

DATA STRUCTURES AND ALGORITHM LAB

PROJECT BASED LEARNING

SUBMITTED TO:

Ms. Sarishty Gupta

SUBMITTED BY:

Mukund Khandelwal (21102134)

Rishabh Jain (21102139)

Mridul Saraswat (21102140)

Aryan Aggarwal (21102151)

TABLE OF CONTENTS

Introduction-

The provided C++ code implements a console-based Sudoku game, offering players an interactive and feature-rich gaming experience. Sudoku, a popular number-placement puzzle, challenges players to fill a 9x9 grid with digits from 1 to 9 in such a way that each row, each column, and each of the nine 3x3 subgrids contain all of the digits without repetition.

The code employs a modular and well-organized structure, making use of various C++ features such as libraries for input/output, time measurement, and data structures like vectors, stacks, and sets. It incorporates functionalities such as saving and loading game states, automatic solving of puzzles, mistake checking, hint provision, and a leaderboard system.

The main function serves as the program's entry point, orchestrating the game flow through a well-defined loop. Players are presented with a menu, allowing them to start a new game, load a saved game, view the leaderboard, or exit the game. The core gameplay is handled by the playGame function, where users interact with the Sudoku puzzle, input moves, and witness the game progress.

The code features a robust leaderboard system that records players' completion times for different difficulty levels. To enhance user experience, it includes functions to display menus, game-over messages, and leaderboards, making the interaction intuitive and engaging.

In summary, this Sudoku game implementation showcases effective code organization, encapsulation of functionalities into modular functions, and utilization of diverse C++ features to create an engaging and feature-rich gaming experience. Players can enjoy the classic Sudoku challenge while benefiting from additional features such as leaderboard tracking and game state persistence.

Features of the Project:

The provided C++ Sudoku game implementation includes a variety of features that enhance the gaming experience. Here's a summary of the main features:

1. **Sudoku Board Display:**
   - Functions to print the Sudoku board on the console.
   - Ability to display the board with mistakes highlighted.
2. **Game Initialization:**
   - Initialization of variables, including the Sudoku puzzle, solution, move stack, elapsed time, and difficulty level.
3. **Save and Load Game:**
   - Functions to save and load the game state, including the puzzle, solution, move history, elapsed time, and difficulty level.
   - Enables players to continue a game from where they left off.
4. **Sudoku Solving and Puzzle Generation:**
   - Ability to solve a Sudoku puzzle automatically.
   - Functionality to generate a new Sudoku puzzle of varying difficulty levels.
5. **Move Validation:**
   - Functions to check if a move is valid before updating the Sudoku board.
6. **Hint Functionality:**
   - Ability to provide hints for the next move based on the puzzle's solution.
7. **Mistake Checking:**
   - Function to identify and highlight mistakes in the player's input.
8. **Leaderboard System:**
   - Loading and saving of a leaderboard from/to file.
   - Displaying the leaderboard with player names and completion times.
   - Evaluation of whether a player's time qualifies for the leaderboard.
9. **User Interface:**
   - Displaying menus for starting a new game, loading a saved game, viewing the leaderboard, and exiting the game.
   - Game-over messages and prompts for player input.
10. **Elapsed Time Tracking:**
    - Measurement of the time taken by the player to complete the Sudoku puzzle.
11. **Auto-Solve Feature:**
    - Capability to automatically solve the Sudoku puzzle for players who may need assistance.

Implementation Details:

1. *Libraries and Namespace Declarations: *

```
cpp
#include <iostream>
#include <chrono>
#include <ctime>
#include <map>
#include <vector>
#include <algorithm>
#include <fstream>
#include <stack>
#include <set>
#include <sstream>
using namespace std;
using namespace chrono;
```

This section includes various C++ libraries for input/output, time measurement, data structures (`map`, `vector`, `stack`, `set`), and file handling.

2. *Constants and Enumerations: *

```
cpp
const int N = 9;
enum Difficulty
{  EASY = 15,
   MEDIUM = 30,
   HARD = 40};
```

Defines the size of the Sudoku board (`N`) and an enumeration (`Difficulty`) representing the difficulty levels along with their corresponding filled cells.

3. *Function Declarations: *

cpp

```
void printBoardWithMistakes(const vector<vector<int>> &board, const vector<vector<int>> &solution, int row, int col);

void printBoard(const vector<vector<int>> &board);

void saveGame(const vector<vector<int>> &puzzle, const vector<vector<int>> &solution,
        const stack<vector<vector<int>>> &moveStack, const Duration &elapsedTime, Difficulty difficulty);

void loadGame(vector<vector<int>> &puzzle, vector<vector<int>> &solution,
        stack<vector<vector<int>>> &moveStack, Duration &elapsedTime, Difficulty &difficulty);

bool isSafe(const vector<vector<int>> &board, int row, int col, int num);

bool solveSudoku(vector<vector<int>> &board);

void generatePuzzle(vector<vector<int>> &puzzle, Difficulty difficulty);

bool isValidMove(const vector<vector<int>> &board, int row, int col, int num);

void provideHint(const vector<vector<int>> &puzzle, const vector<vector<int>> &solution);

set<pair<int, int>> checkMistakes(const vector<vector<int>> &board, const vector<vector<int>> &solution);

void loadLeaderboard(map<string, double> &leaderboard, Difficulty difficulty);

void saveLeaderboard(const map<string, double> &leaderboard, Difficulty difficulty);

void autoSolveSudoku(vector<vector<int>> &board);

string getDifficultyName(Difficulty difficulty);

string getDifficultyFileName(Difficulty difficulty);
```

These are function declarations. They define the interface for various functionalities of the Sudoku game.

4. *Print Board Functions: *

cpp

```
void printBoard(const vector<vector<int>> &board);
void printBoardWithMistakes(const vector<vector<int>> &board, const vector<vector<int>> &solution, int row, int col);
```

These functions print the Sudoku board, with the second function highlighting mistakes in red.

5. *Save and Load Game Functions: *

cpp

```
void saveGame(const vector<vector<int>> &puzzle, const vector<vector<int>> &solution,
              const stack<vector<vector<int>>> &moveStack, const Duration &elapsedTime, Difficulty difficulty);

void loadGame(vector<vector<int>> &puzzle, vector<vector<int>> &solution,
              stack<vector<vector<int>>> &moveStack, Duration &elapsedTime, Difficulty &difficulty);
```

These functions save and load the game state, including the puzzle, solution, move history, elapsed time, and difficulty level.

6. *Sudoku Solving and Puzzle Generation Functions: *

cpp

```
bool isSafe(const vector<vector<int>> &board, int row, int col, int num);
bool solveSudoku(vector<vector<int>> &board);
void generatePuzzle(vector<vector<int>> &puzzle, Difficulty difficulty);
```

These functions provide the core Sudoku solving and puzzle generation logic.

7. *Move Validation and Hint Functions: *

cpp

bool isValidMove(const vector<vector<int>> &board, int row, int col, int num);

void provideHint(const vector<vector<int>> &puzzle, const vector<vector<int>> &solution);

These functions check if a move is valid and provide hints for the next move.

8. *Mistake Checking Function: *

cpp

set<pair<int, int>> checkMistakes(const vector<vector<int>> &board, const vector<vector<int>> &solution);

This function identifies mistakes in the player's input.

9. *Leaderboard Functions: *

cpp

void loadLeaderboard(map<string, double> &leaderboard, Difficulty difficulty);

void saveLeaderboard(const map<string, double> &leaderboard, Difficulty difficulty);

These functions load and save the leaderboard data from/to files.

10. *Auto-Solve Function: *

cpp

void autoSolveSudoku(vector<vector<int>> &board);

This function automatically solves the Sudoku puzzle.

11. *Utility Functions: *

```
cpp
string getDifficultyName(Difficulty difficulty);
string getDifficultyFileName(Difficulty difficulty);
```

These functions return the name and file name corresponding to the given difficulty level.

12. *Main Function: *

```
cpp
int main()
{
    // ... (see the main function for details)
}
```

The main function is where the game logic is implemented. It takes care of player input, game flow, and interaction.

In summary, the code implements a Sudoku game with features like saving/loading, hinting, mistake checking, auto-solving, and a leaderboard. The game is played through the console, and the player's progress and leaderboard data are stored in files.

Result and Analysis:

Invalid Move-

```
Fill in the empty cells. Enter row, column, and value (separated by spaces) to make a move.
. . 3 | 4 5 6 | 7 8 9
. 5 6 | . 8 9 | 1 2 3
. 8 9 | 1 2 3 | 4 5 6
---------------------
. 1 . | 3 6 . | 8 9 7
3 6 5 | 8 . 7 | 2 1 4
. 9 7 | 2 . 4 | 3 6 5
---------------------
5 3 1 | 6 . 2 | 9 7 .
6 4 2 | 9 7 8 | 5 3 .
. 7 8 | 5 3 1 | 6 4 2
Do you want to resume a saved game? (y/n): n
Enter your move (row col value, -1 to undo, 0 for hint, -2 to auto-solve, -3 to save): 1 1 3
Invalid move. Try again.
```

Wrong Move-

```
Do you want to resume a saved game? (y/n): n
Enter your move (row col value, -1 to undo, 0 for hint, -2 to auto-solve, -3 to save): 1 1 2
2 . 3 | 4 5 6 | 7 8 9
. 5 6 | . 8 9 | 1 2 3
. 8 9 | 1 2 3 | 4 5 6
---------------------
. 1 . | 3 6 . | 8 9 7
3 6 5 | 8 . 7 | 2 1 4
. 9 7 | 2 . 4 | 3 6 5
---------------------
5 3 1 | 6 . 2 | 9 7 .
6 4 2 | 9 7 8 | 5 3 .
. 7 8 | 5 3 1 | 6 4 2
Enter your move (row col value, -1 to undo, 0 for hint, -2 to auto-solve, -3 to save):
```

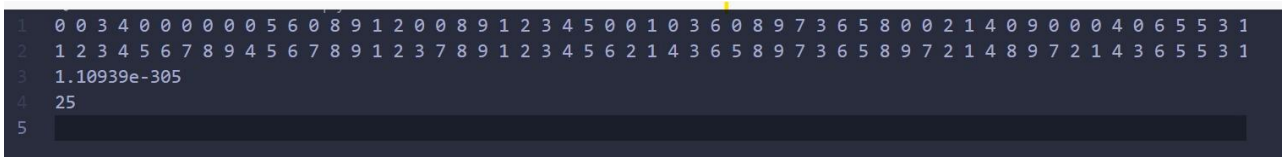
Hint-

```
Enter your move (row col value, -1 to undo, 0 for hint, -2 to auto-solve, -3 to save): 0 0 0
Hint: Place 8 at position (2, 5).
```

Auto Solve-

```
Enter your move (row col value, -1 to undo, 0 for hint, -2 to auto-solve, -3 to save): 0 0 -2
Auto-Solving...
1 2 3 | 4 5 6 | 7 8 9
4 5 6 | 7 8 9 | 1 2 3
7 8 9 | 1 2 3 | 4 5 6
---------------------
2 1 4 | 3 6 5 | 8 9 7
3 6 5 | 8 9 7 | 2 1 4
8 9 7 | 2 1 4 | 3 6 5
---------------------
5 3 1 | 6 4 2 | 9 7 8
6 4 2 | 9 7 8 | 5 3 1
9 7 8 | 5 3 1 | 6 4 2
Do you want to play again? (y/n):
```

Saved Text File-



Leaderboard -

Conclusion:

In conclusion, the C++ implementation of the Sudoku game showcases a well-structured and feature-rich design, offering players an engaging and interactive puzzle-solving experience. The code leverages a variety of C++ features, including standard libraries for input/output, time measurement, and data structures such as vectors, stacks, and sets.

The implementation covers a broad range of functionalities, including Sudoku board display, game initialization, saving and loading game states, Sudoku solving, puzzle generation, move validation, hint provision, mistake checking, leaderboard management, and an auto-solve feature. These features contribute to a comprehensive gaming experience that goes beyond the basic requirements of a Sudoku puzzle.

The code's modularity and organization are noteworthy, with distinct functions handling specific aspects of the game. This design choice enhances readability, maintainability, and extensibility, making it easier to understand and build upon the existing codebase.

Additionally, the incorporation of a leaderboard system adds a competitive and social element to the game, allowing players to track their completion times and compete for top positions. The inclusion of features such as auto-solving and saving/loading game states enhances user convenience and flexibility.

Overall, the Sudoku game implementation reflects effective coding practices, thoughtful design decisions, and a commitment to providing players with a robust and enjoyable gaming experience. Whether a player is seeking a classic Sudoku challenge or exploring additional features, this implementation caters to a diverse range of preferences within the framework of the timeless puzzle.

Division of Work among Team Members:

The project was a collaborative effort with each team member contributing equitably to its success. We adopted a systematic approach, ensuring that responsibilities were distributed evenly based on individual strengths and interests. Each member actively participated in algorithmic development, code implementation, and debugging processes. The division of work encompassed tasks such as refining the backtracking algorithm, implementing visualization features, and optimizing code efficiency.
.

1. Aryan
• Board Display and Printing
• Game State Management

2. Mukund and RJ
- Sudoku Solving and Puzzle Generation
- Move Validation and Hint Functionality
- Mistake Checking
- Leaderboard Management

3. Mridul
- Auto-Solve Functionality
- Main Function and Game Flow

References

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.
- Knuth, D. E. (2011). The Art of Computer Programming, Volume 4, Fascicle 0: Introduction to Combinatorial Algorithms and Boolean Functions. Addison-Wesley.
- GeeksforGeeks. (https://www.geeksforgeeks.org/)