***DA213: Python Programming Laboratory***

***Course Project***

***Title: Sudoku Game & Solver***

***Guidance By: Dr. Teena Sharma***

***(Asst. Prof. of MFSDSAI)***

***Teaching Asst.: Pallapu Mohan Krishna***

***Prakhar Kumar Sonkar***

***Group Name: Lockdown Group***

***Group Members: Sahil Raj***

***(220150018)***

***Vishal***

***(220150029)***

***Abstract:***  
  
The Sudoku game project aims to develop a digital implementation of the popular Sudoku puzzle game using Python programming language and pygame library. The project includes features such as a graphical user interface (GUI) for interactive gameplay, a Sudoku puzzle generator with varying levels of difficulty, and a backtracking algorithm for solving puzzles. The game allows players to solve pre-generated puzzles or input their own custom puzzles for solving. The project also includes a solver component that can automatically solve any valid Sudoku puzzle. Overall, the Sudoku game project provides an enjoyable and challenging gaming experience for Sudoku enthusiasts while showcasing fundamental programming concepts such as algorithm design, data structures, and graphical user interface development.

***Table of Contents:***

1. Introduction

- Background Information

- Objectives & Goals

- Scope

- Limitations

2. Literature Review

- Review of Relevant Literature and Existing Solutions  
- Explanation of How the Project Builds Upon Existing Sudoku Game Projects

3. Mathematical Foundations

4. Methodology

- Description of the Approach Used

- Several algorithms, technologies, and tools

5. Results

- One player game

- Solution for given sudoku’s input

6. Discussions

7. Performance Metrics

8. Conclusion

- Summary

- Achievements

- Suggestions for future work or improvement

***Introduction***

***(a) Background Information:***

Sudoku is a logic-based number puzzle game that became popular in the late 20th century. It is played on a 9 x 9 grid, divided into nine 3 x 3 subgrids called "regions". The goal of the game is to fill the grid with numbers from 1 to 9, ensuring that each row, each column, and each region contains all the numbers exactly once.  
  
The Sudoku game project was conceived to provide a digital platform for Sudoku enthusiasts to enjoy the game and challenge themselves with various levels of difficulty. The project also serves as an educational tool for learning programming concepts such as algorithm design, data structures, and graphical user interface development.  
  
By implementing a Sudoku game, developers can explore various algorithms for generating puzzles, solving puzzles, and validating solutions. The project can also be extended to include features such as timer functionality, scoring systems, and multiplayer modes, enhancing the overall gameplay experience.  
  
Overall, the Sudoku game project offers a fun and engaging way to test one's logical thinking and problem-solving skills while providing a platform for exploring programming concepts in a practical and interactive manner.

***(b) Objectives & Goals:***  
1. Develop a fully functional Sudoku game using Python programming language and pygame library.  
2. Implement a graphical user interface (GUI) for interactive gameplay, allowing users to easily input and solve Sudoku puzzles.  
3. Create a Sudoku puzzle generator that can generate puzzles with varying levels of difficulty.  
4. Design a backtracking algorithm to solve Sudoku puzzles automatically, providing hints and solutions to players.  
5. Allow players to choose from different difficulty levels (easy, medium, hard,expert) and the time taken to solve the puzzle.  
6. Ensure the game's user interface is intuitive and visually appealing, enhancing the overall user experience.  
7. Provide options for players to customize and input their own Sudoku puzzles for solving.  
8. Include error checking functionality to prevent players from entering invalid or incorrect solutions.  
9. Optimize the game's performance and code efficiency to ensure smooth gameplay even with complex puzzles.  
10. Document the development process, including the algorithms used, challenges faced, and lessons learned, to serve as a reference for future projects and learning opportunities.

***(c) Scope:***

1. Implement a Sudoku game with a graphical user interface (GUI) using Python and pygame library.  
2. Develop a Sudoku puzzle generator with varying levels of difficulty.  
3. Design a backtracking algorithm to solve Sudoku puzzles automatically.  
4. Include features such as error checking, hint functionality, and timer.  
5. Allow players to input custom puzzles for solving.  
6. Provide a visually appealing and user-friendly interface.

***(d) Limitations:***

1. The game will be limited to a single-player experience.  
2. The GUI may not be optimized for all screen sizes and resolutions.  
3. The puzzle generator may not always generate puzzles with a unique solution.  
4. The backtracking algorithm may not be optimized for solving extremely complex puzzles efficiently.  
5. The game may lack advanced features found in commercial Sudoku games, such as multiplayer modes or extensive puzzle statistics.  
6. The game's performance may vary depending on the complexity of the puzzle and the system's resources.  
7. The project may not cover all possible Sudoku variations or rule sets, focusing mainly on the classic 9x9 Sudoku puzzle.

***Literature Review***

***(a) Review of Relevant Literature and Existing Solutions:***

1. Sudoku Solving Algorithms: Various algorithms have been proposed for solving Sudoku puzzles, including backtracking, constraint propagation (e.g., the Sudoku-X algorithm), and brute-force methods. These algorithms form the basis for developing the puzzle solver component of the Sudoku game.  
  
2. GUI Development: Studies on GUI design principles and user interface development in Python can provide insights into creating a visually appealing and user-friendly interface for the Sudoku game.  
  
3. Puzzle Generation Techniques: Research on Sudoku puzzle generation techniques, such as deterministic and stochastic methods, can guide the development of the puzzle generator component to ensure the puzzles are challenging yet solvable.  
  
4. Error Checking and Hint Systems: Literature on error checking algorithms and hint systems in puzzle games can inform the implementation of features that help players identify and correct mistakes while solving Sudoku puzzles.  
  
5. Game Design and User Experience: Studies on game design principles and user experience (UX) design can provide guidance on creating engaging gameplay, intuitive controls, and rewarding feedback mechanisms in the Sudoku game.  
  
6. Optimization Techniques: Research on algorithm optimization techniques, such as pruning strategies in backtracking algorithms, can help improve the performance of the Sudoku puzzle solver, especially for large and complex puzzles.  
  
By reviewing the existing literature and solutions in these areas, the Sudoku game project can benefit from established practices and techniques, leading to a more robust and enjoyable gaming experience for players.

***(b) Explanation of How the Project Builds Upon Existing Sudoku Game Projects:***  
  
1. Enhanced User Interface: Our project aims to provide a more visually appealing and user-friendly interface compared to existing Sudoku game implementations. By using Python and the pygame library, we can create a dynamic and interactive GUI that enhances the overall user experience.  
  
2. Customization Options: Unlike some existing Sudoku games that offer limited customization options, our project allows players to input their own custom puzzles for solving. This feature adds a personal touch to the gameplay and provides a unique challenge for players.

3. Efficient Puzzle Generation: While many Sudoku games use simple puzzle generation algorithms, our project aims to implement a more sophisticated puzzle generator that can create puzzles with varying levels of difficulty. This ensures that players are constantly challenged and engaged.  
  
4. Advanced Solver Algorithm: Our project includes a backtracking algorithm for solving Sudoku puzzles automatically. This algorithm is optimized for efficiency and can solve even the most complex puzzles quickly, providing hints and solutions to players when needed.  
  
5. Error Checking and Hint System: To enhance the gameplay experience, our project includes features such as error checking and hint functionality. These features help players identify and correct mistakes, ensuring a smooth and enjoyable gaming experience.  
  
Overall, our project builds upon existing Sudoku game projects by offering a more interactive and customizable gameplay experience, with advanced features that cater to both casual players and Sudoku enthusiasts.

***Mathematical Foundation***

The Sudoku game project is grounded in several mathematical concepts and principles, including:  
  
1. Combinatorics: The number of possible Sudoku grids is a combinatorial problem. Combinatorics helps in understanding the total number of valid Sudoku grids and the complexity of generating new puzzles.  
  
2. Graph Theory: Sudoku puzzles can be represented as a graph, where cells are nodes and constraints (row, column, and region constraints) are edges. Graph theory provides a framework for understanding the relationships between cells and for developing solving algorithms.  
  
3. Constraint Satisfaction Problems (CSPs): Sudoku can be formulated as a CSP, where the goal is to assign values to variables (cells) subject to constraints (row, column, and region constraints) to satisfy a set of predefined conditions. CSP theory provides algorithms for solving Sudoku puzzles.  
  
4. Algorithm Design: The backtracking algorithm is commonly used to solve Sudoku puzzles. Understanding algorithmic concepts such as recursion, branching, and pruning is essential for implementing an efficient solver.  
  
5. Number Theory: Sudoku involves assigning numbers (1 to 9) to cells such that each row, column, and region contains each number exactly once. Number theory concepts, such as divisibility rules, can be applied to ensure the validity of solutions.  
  
6. Probability and Statistics: While not as central, probability and statistics can be used to analyze the randomness of puzzle generation algorithms and the distribution of clues in generated puzzles.  
  
7. Linear Algebra: Although not directly applied in typical Sudoku solving algorithms, linear algebra concepts can be used in more advanced Sudoku variations that involve additional constraints or dimensions.  
  
By leveraging these mathematical foundations, the Sudoku game project can develop efficient solving algorithms, puzzle generation techniques, and validation methods, providing players with challenging and engaging gameplay.

***Methodology***

***(a) Description of the Approach Used to Complete the Project:***  
1. Requirement Analysis: The project started with a thorough analysis of the requirements, including the desired features, user interface design, and functional specifications. This phase also involved researching existing Sudoku games to understand their strengths and weaknesses.  
  
2. Design Phase: Based on the requirements, a detailed design was created, outlining the architecture of the Sudoku game. This included designing the graphical user interface (GUI), puzzle generation algorithm, solver algorithm, and error checking mechanisms. Consideration was given to the user experience (UX) and the overall visual appeal of the game.  
  
3. Implementation: The Sudoku game was implemented using Python programming language and the pygame library. The GUI was created to allow players to interact with the game, input custom puzzles, and receive hints. The puzzle generator was designed to generate puzzles with varying levels of difficulty, and the solver algorithm was implemented using a backtracking approach to solve puzzles automatically.  
  
4. Testing: The Sudoku game underwent rigorous testing to ensure that all features worked as intended and that the game was free of bugs and errors. This included testing the GUI, puzzle generation, solver algorithm, error checking, and overall game logic.  
  
5. Optimization: The code was optimized for efficiency and performance, especially the solver algorithm, to ensure that the game could handle complex puzzles smoothly and without delay.  
  
6. Documentation: Throughout the development process, detailed documentation was maintained, including design decisions, algorithms used, challenges faced, and lessons learned. This documentation serves as a reference for future projects and learning opportunities.  
  
7. Feedback and Iteration: Feedback from users and stakeholders was gathered to identify areas for improvement. Based on this feedback, iterations were made to enhance the game's features, performance, and overall user experience.  
  
8. Finalization: The Sudoku game project was finalized, ensuring that it met all requirements and goals. The project was prepared for presentation and submission, including preparing user documentation for playing the game.

***(b) For the Sudoku game project, several algorithms, technologies, and tools were utilized:***  
1. Backtracking Algorithm: Used to solve Sudoku puzzles, the backtracking algorithm systematically tries all possible numbers in empty cells until a solution is found. If a conflict is detected (e.g., a number violates the Sudoku rules), it backtracks and tries a different number.  
  
2. Puzzle Generation Algorithm: A custom algorithm was developed to generate Sudoku puzzles with varying levels of difficulty. The algorithm ensures that the generated puzzles have a unique solution and are solvable by applying logical reasoning.  
  
3. Python Programming Language: Python was chosen as the primary programming language for its simplicity, readability, and availability of libraries for GUI development(pygame).

4. Pygame Library: Used for creating the graphical user interface (GUI) of the Sudoku game. Pygame provides a simple way to draw shapes and images on the screen, making it suitable for creating the Sudoku grid and displaying numbers. And creating the GUI elements of the Sudoku game, such as buttons, labels, and text entry fields.  
  
5. Integrated Development Environment (IDE): An IDE such as Spyder or Visual Studio Code was used for writing, debugging, and running the Python code. IDEs provide features like syntax highlighting, code completion, and debugging tools that streamline the development process.  
  
6. Documentation Tools: Tools like MS Word and LaTeX were used for documenting the project, including writing the project report and creating user documentation. These tools provide a structured way to format and present information.  
  
Overall, the combination of these algorithms, technologies, and tools facilitated the development of a functional and user-friendly Sudoku game with a variety of features and capabilities.

***Results***

***(a) One player game:***

(i) There are four difficulty levels: Easy, Medium, Hard & Expert.

(ii) Then sudoku is generate within grid with timer on the basis of difficulty level selected.

(iii) If user’s input right answer, then the particular box turns into green & if user’s input wrong answer, then

the particular box turns into red.

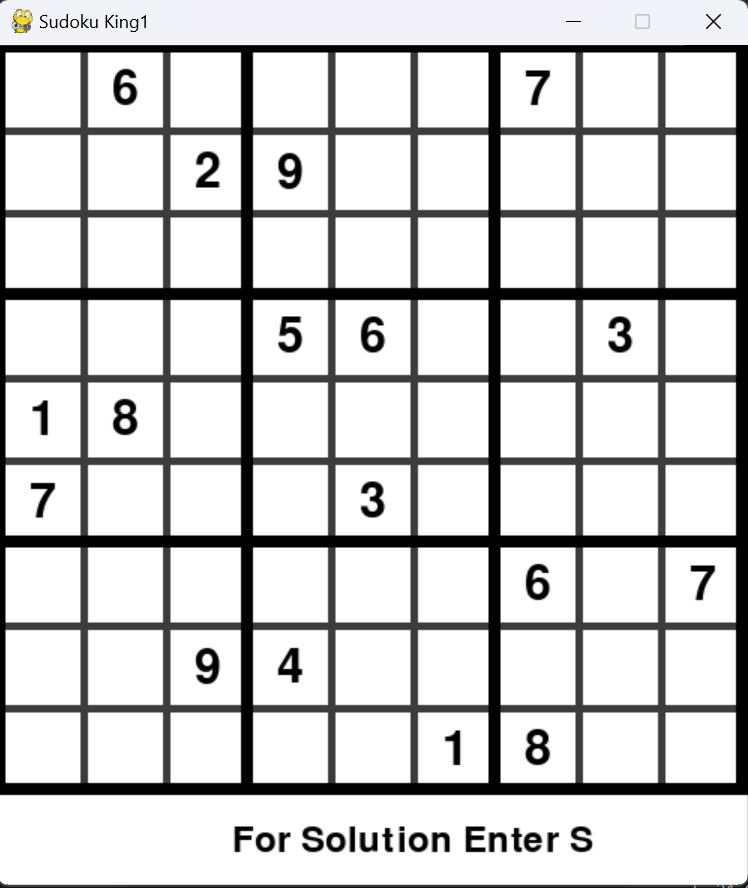
(iv) After three wrong input, the game will over.

(v) User can get hint for a particular box by pressing H button, then the particular box turns into yellow.

(vi) If user wants to know the solution of the whole sudoku, then he/she will get the solution by pressing C

Button(even after lost the sudoku).

***(b) Solution for given sudoku’s input:***

******

******

***Discussions***

For the discussion part of your Sudoku game project, you could include various aspects of the project such as:  
  
1. Introduction: Briefly describe the purpose and goals of your project, including the motivation behind creating a Sudoku game.  
  
2. Game Mechanics: Explain how the game works, including the rules of Sudoku and any additional features or variations you've implemented.  
  
3. Implementation Details: Discuss the technologies and programming languages used to develop the game, as well as any libraries or frameworks utilized.  
  
4. Challenges Faced: Describe any challenges or obstacles you encountered during the development process and how you overcame them.  
  
5. Lessons Learned: Reflect on what you learned from working on this project, including new programming techniques, problem-solving skills, or insights into game development.  
  
6. Future Improvements: Share ideas for future enhancements or features you would like to add to the game.  
  
7. Conclusion: Summarize the key points of your discussion and the overall outcome of the project.

***Performance Metrics***

For a Sudoku game project, you can consider several performance metrics to evaluate its efficiency and user experience. Here are some key metrics you might consider:  
  
1. Game Load Time: Measure the time it takes for the game to load initially. This includes any assets such as images, sounds, or game data.  
  
2. Game Start Time: Measure the time from when the user initiates starting a new game to when the game board is displayed and ready for play.  
  
3. Game Play Time: Measure the average time users spend playing a game. This can help assess the game's engagement level.  
  
4. Algorithm Efficiency: If you have implemented a solver or generator for Sudoku puzzles, measure the time it takes for these algorithms to complete.  
  
5. Memory Usage: Monitor the game's memory consumption, especially during intense operations like puzzle generation or solving, to ensure it stays within reasonable limits.  
  
6. User Interaction Metrics: Track how users interact with the game, such as how quickly they complete puzzles, how often they use hints or undo features, and how many puzzles they solve in a single session.  
  
7. Error Handling: Measure how the game handles incorrect inputs or invalid moves, including how quickly errors are detected and how they are communicated to the user.

8. User Satisfaction: Consider implementing surveys or feedback mechanisms to gather user opinions on the game's performance, usability, and overall enjoyment.  
  
9. Compatibility: Test the game on different devices and platforms to ensure compatibility and measure any performance differences between them.  
  
10. Scalability: If your game supports multiple difficulty levels or puzzle sizes, measure how well it scales in terms of performance and user experience across these variations.  
  
By tracking these metrics, you can gain insights into the performance and user experience of your Sudoku game and identify areas for improvement.

***Conclusion***

***(a) Summary:***  
The Sudoku game project aimed to create an interactive and engaging game based on the popular Sudoku puzzle. The project was implemented using Python with the Turtle graphics library for the graphical user interface.  
  
**Key Features:**  
1. Implemented a Sudoku game engine capable of generating new puzzles and solving existing ones.  
2. Designed an intuitive user interface with features such as input validation, error detection, and hint functionality.  
3. Provided different difficulty levels to cater to players of varying skill levels.  
4. Implemented a solver using backtracking algorithm to provide hints to the player.  
5. Utilized Turtle graphics for drawing the game board and user interface elements.

**Challenges:**  
1. Designing an efficient algorithm for puzzle generation and solving.  
2. Ensuring a smooth and responsive user interface.  
3. Handling user inputs and validating them against Sudoku rules.

***(b) Achievements:***

1. Successfully implemented a fully functional Sudoku game with an intuitive user interface.  
2. Achieved the project goals of providing a fun and challenging gaming experience for players.

***(c) Future Improvements:***

1. Implementing additional features such as saving and loading game states.  
2. Enhancing the user interface with more interactive elements and animations.  
3. Adding multiplayer functionality for competitive gameplay.  
Overall, the Sudoku game project was a valuable learning experience in game development and algorithm design, providing a solid foundation for future projects in this field.