DE LA SALLE UNIVERSITY - MANILA

SudokuPro

A Term Project

Presented to Mr. Ramon Stephen L. Ruiz

In Partial Fulfillment of the

Requirements for the Course Programming Logic and Design Laboratory (LBYCPA1)

---

by

CRUZ, Jose Andres A.

ITCHON, Patrice Tracy P.

YANO, Alexis Mae O.

EQ1

April 2023

I. Introduction

A. Background of the Study

Sudoku is a popular puzzle game that has gained significant attention in recent years due to its ability to improve cognitive skills such as memory, logical thinking, and problem-solving. The game was first introduced in Japan in the late 1980s and has since gained a worldwide following (Li et al., 2015).

Python is a popular programming language that has been used to develop many applications, including games. With its simplicity and ease of use, Python has become the language of choice for many developers looking to create games, including Sudoku (Pozza et al., 2021).

In this research project, the aim is to develop a Sudoku game in Python that will provide users with an engaging and challenging gaming experience. The game will include a user interface, game logic, game difficulty level, and the use of algorithms to solve puzzles. Previous research has shown that playing Sudoku can improve cognitive skills such as memory, logical thinking, and problem-solving (Li et al., 2015). By developing a Sudoku game in Python, its hope is to contribute to the growing body of research on the benefits of playing Sudoku and provide users with an enjoyable and stimulating gaming experience.

B. Problem Statement

1. How to design and implement a user interface for a Sudoku game in Python?
2. How to implement the game logic for a Sudoku game in Python?
3. How to generate Sudoku puzzles with varying levels of difficulty in Python?
4. How to use algorithms to solve Sudoku puzzles in Python?
5. What is the user experience of playing a Sudoku game in Python?

C. Objectives

C1. General Objectives

The general objective of this research project is to develop a Sudoku game in Python that will provide users with an engaging and challenging gaming experience while contributing to the growing body of research on the benefits of playing Sudoku as a cognitive training tool.

C2. Specific Objectives

1. To design and implement a user interface for a Sudoku game in Python that is intuitive and user-friendly.
2. To implement the game logic for a Sudoku game in Python that adheres to the rules of the game and allows for efficient manipulation of the game board.
3. To generate Sudoku puzzles with varying levels of difficulty in Python to provide users with a challenging gaming experience.
4. To use algorithms to solve Sudoku puzzles in Python to enable the game to validate user solutions and provide hints or solutions to users who get stuck.
5. To evaluate the user experience of playing a Sudoku game in Python through user testing and feedback.

D. Significance of the Project

The significance of this study lies in its contribution to the growing body of research in the academe and also on the benefits of playing Sudoku and its potential as a cognitive tool. By developing a Sudoku game in Python, it is great how users find an enjoyable and stimulating game experience while also contributing to the development of tools for cognitive training.

Furthermore, this study can serve as a reference for future students and developers looking to create beginner games in Python. The methods and techniques used in this

project can be applied to other games or applications, providing valuable insights for future researchers, students, and developers.

II. Review of Related Literature

The origins of Sudoku may be found in a puzzle game from the 18th century called "Latin Squares," and the first number problems initially appeared in French newspapers in 1895. The game gained popularity in Japan in 1984, where it was given the name "Sudoku," which translates to "the digits are limited to one occurrence" in Japanese, but the contemporary version of Sudoku that is well-known today was created by Howard Garns in 1979 and published in Dell Pencil Puzzles and Word Games magazine under the name "Number Place." More than 600,000 Sudoku magazines are distributed each month in Japan, and the World Sudoku Championships, which are conducted every year, have made sudoku a worldwide sensation.

A New Zealand judge by the name of Wayne Gould is credited with reintroducing Sudoku to the West in 1997 while on vacation in Tokyo. Gould developed a computer software that could produce Sudoku problems after becoming a Sudoku lover. Sudoku puzzles were first published by The Times of London in 2004, and the following year, publications like The Conway Daily Sun began to feature them. Sudoku puzzles swiftly acquired popularity in the United States as well. The attractiveness of Sudoku is partly due to its simplicity and easily learnable principles for players of all ages. Additionally, Sudoku may be played by anybody anywhere in the globe without the requirement for translation because of its number-based style. The appeal of sudoku is partly ascribed to its capacity to appease people's ingrained sense of order and their need for a stimulating and difficult logic challenge in the fast-paced world of today.

Popular puzzle game Sudoku depends on logical deduction rather than mathematics. The fundamental tactic entails forcing entries into cells with only one option while filling in all vacant cells with potential entries using the "forced entry" method. Another tactic is to concentrate on a number and a row, column, or block, then rule out other solutions using the One Rule. The use of pairs or triples of cells with few

possibilities is a more sophisticated strategy that can assist remove possibilities in nearby cells. Players may make informed estimates and keep employing the aforementioned tactics if no entries are required. Players must go back and undo past movements if a contradiction develops until a solution without contradictions is discovered.

According to new research in the International Journal of Geriatric Psychiatry, adults over 50 may benefit cognitively by doing word and numerical puzzles like sudoku and crosswords. Data from about 19,100 participants were evaluated for the study, and it was shown that those who solved puzzles did better on tests of attention, memory, and reasoning. The benefits were especially noticeable in performance speed and accuracy, with problem solvers displaying brain function comparable to people who were 8–10 years younger.

Further research is required to determine the long-term effects of puzzle involvement and intensity because the study did not demonstrate that puzzles can prevent dementia in later life. However, the results are consistent with other studies suggesting that solving puzzles on a regular basis may help older people maintain superior cognitive performance. To further understand the connection between puzzles and older persons' brain health, the researchers want to monitor the participants over time and evaluate variables including problem difficulty and duration of participation.

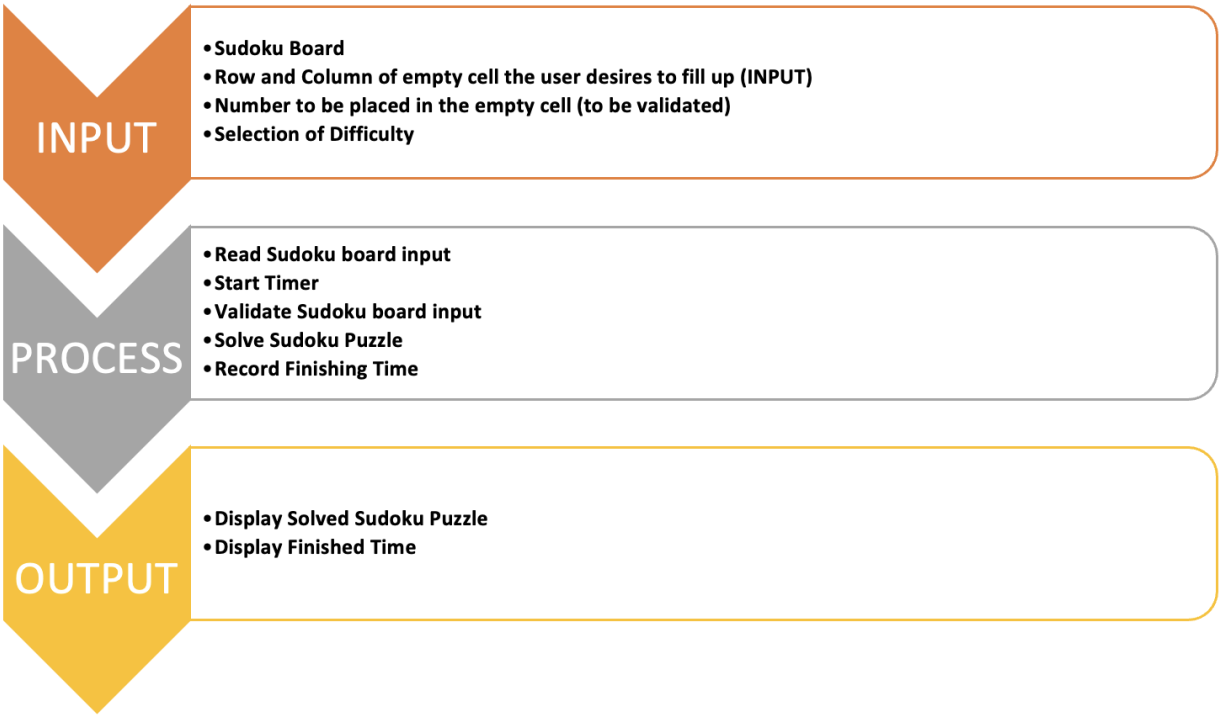
Regardless of age, playing Sudoku often can have a variety of positive effects on the brain. While it may assist the elderly avoid Alzheimer's disease, the most prevalent form of dementia, it can also aid children with their focus and memory. Sudoku demands creative problem-solving, strategic thinking, and reasoning, all of which can improve attention and refocusing abilities. Through logical reasoning, it also stimulates the intellect and enhances mathematical abilities. Additionally, Sudoku can help with decision-making and time management abilities since it teaches players to make decisions on time and take action on them quickly. Additionally, completing a challenging problem may make you feel happy and accomplished, which can increase

your overall happiness. Sudoku is a fun and advantageous game for brain health since it may keep the brain busy and lower the risk of Alzheimer's.

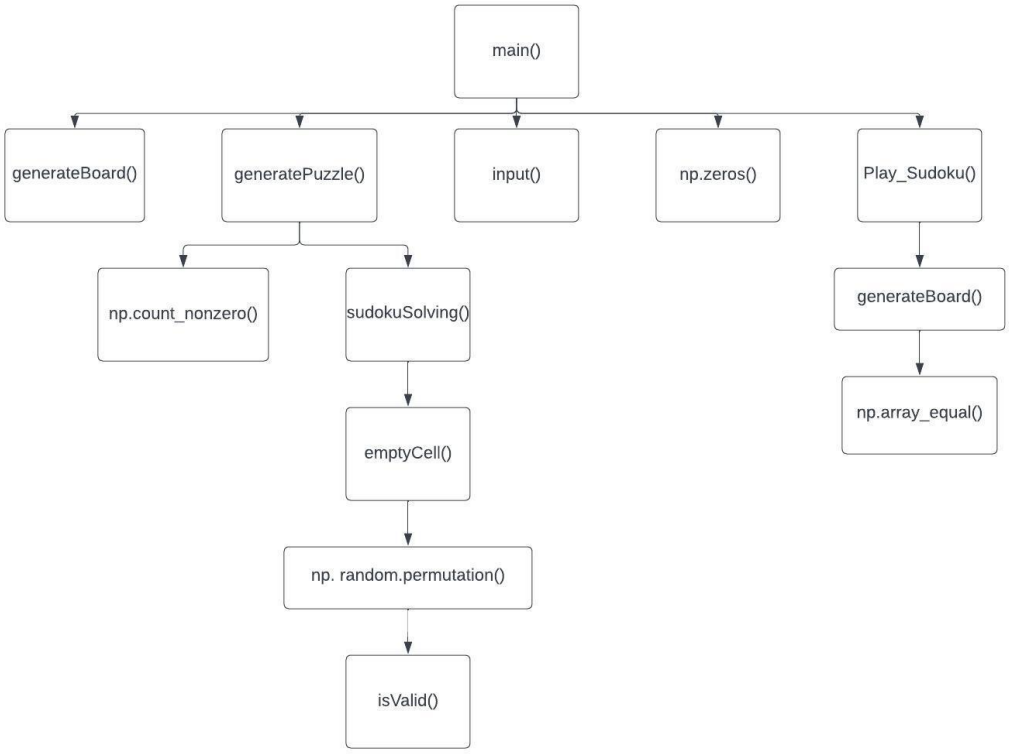
Numerous real-time applications may be made of the extremely logical game of sudoku. First of all, it promotes mental clarity, allowing players to think more clearly and effectively, which can lead to superior problem-solving abilities in actual life. Second, Sudoku has developed into a competitive sport with players and grid builders making a career off of the game and its advantages. Thirdly, Sudoku algorithms are employed in Artificial Intelligence to teach robots, assisting programmers in comprehending and customizing human behavior. Sudoku grids also have a mathematical connection; certain grids are converted into coloring grids to resolve mathematical dilemmas. Sudoku grids are occasionally used in spyware methods like Steganography to conceal hidden communications, with a key placed behind the problem to reveal the solution. As a result, Sudoku has many uses outside of being merely a game, including job prospects, solving mathematical problems, and even using spyware and artificial intelligence.

III. Methodology

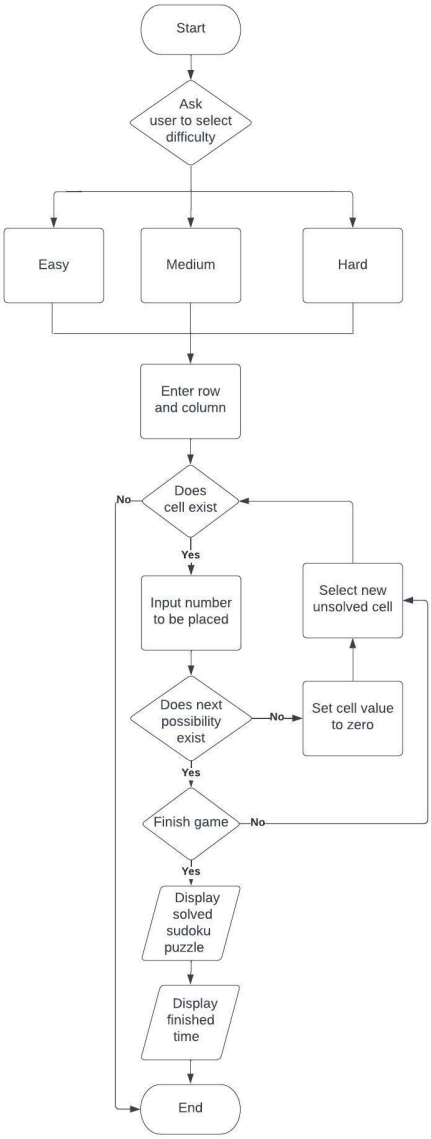
A. Conceptual Framework - IPO Chart



B. Hierarchy Chart



C. Flowchart



D. Pseudocode

START
SET next_action to "play again"
WHILE next_action is "play again" DO
INPUT sudoku_board
VALIDATE sudoku_board
WHILE sudoku_board is not valid DO
DISPLAY error message
INPUT sudoku_board
VALIDATE sudoku_board

END WHILE
SOLVE sudoku_board
RECORD finishing_time
UPDATE leaderboard with user_name and finishing_time
DISPLAY solved sudoku_board
DISPLAY leaderboard
PROMPT for next action (play again or exit)
IF next action is "play again" THEN
CONTINUE
ELSE
SET next_action to "exit"
ENDIF
END WHILE
END

IV. Results

The program is specifically designed to create Sudoku puzzles with different levels of difficulty. It utilizes a complex algorithm to generate these puzzles, ensuring that they are challenging and engaging for the user. The program provides an interactive interface for the user to input their answers and check if they are correct or not. Once the puzzles are created, the user has the option to play them and try to solve them.

V. Discussion of Results

It generates a puzzle of varying difficulty levels for the player to solve. The program uses the backtracking algorithm to solve the puzzle and randomly removes numbers from the solution to generate an unsolved puzzle.

The program starts by asking the user to choose the difficulty level: Easy, Medium, Hard or Daily Challenge. If Daily Challenge is chosen, a new random seed is generated based on the current date to provide a unique challenge each day. The program then generates an empty 9x9 Sudoku board and uses the backtracking algorithm to solve it. The solved board is then used to generate a new puzzle of the chosen difficulty level. The program removes random numbers from the solution and ensures that the puzzle has only one unique solution.

The player can enter the row, column, and number to fill in the empty cell. If the entered number matches the number in the solved board, the cell is filled in the unsolved board, and the updated board is displayed. If the entered number is incorrect, the program displays the updated board without making any changes. The game ends when the player

correctly fills in all the cells in the unsolved board. Once the game is over, the program displays the solved board and thanks the player for playing.

VI. Analysis, Conclusion, and Future Directives

The project is a version of the well-known Sudoku game. It allows users to choose the level of difficulty, generates a puzzle, and allows them to play the game by inserting numbers until they have solved the puzzle.

The code begins with the appropriate imports of the datetime, numpy, random, and time modules. The primary function begins by asking the user to select the level of difficulty. If the user enters 4, the code creates a daily challenge depending on the current date. The board is then solved by creating a board of zeros and passing it to the function sudokuSolving. If a solved board is achieved, it is copied and handed to the generatePuzzle function, which creates a puzzle with the chosen difficulty level. The generateBoard function displays the created puzzle and creates a copy of the unsolved board for the user to play with.

The generateBoard function is responsible for displaying the board. It takes the board array as an input and loops through its elements. It prints a vertical line followed by a space or a number, depending on the value in the board array. If the current column number is divisible by 3, a vertical line is printed to separate each 3x3 sub-grid. If the current row number is divisible by 3, a horizontal line is printed to separate each 3x3 sub-grid.

The presentation of the board is done via the generateBoard function. It runs through the items of the board array as an input. Depending on the value in the board array, it displays a vertical line followed by a space and either a number or a space. Each 3x3 sub-grid is separated by a vertical line if the current column number is divisible by 3. Each 3x3 sub-grid is separated by a horizontal line if the current row number is divisible by 3.

The Play_Sudoku function enables users to play the game by entering numbers into blank cells until they have finished the sudoku. It accepts both the solved and unsolved boards as input and loops until the user either completes the problem or enters 10 to end the game. The user is prompted to specify the row, column, and number to insert. It alerts the user if the site is already occupied. If the number entered is accurate, it updates the unsolved board and prints the updated board. It alerts the user and prints the previous board if the number entered is wrong. It prints a message of congratulations if the user is the one who has finished the problem.

The isValid function examines if placing the number at the given place would be against Sudoku's rules by taking the board, row, column, and number as inputs. It determines whether the supplied number is present in the same row, column, or 3x3 sub-grid and returns False if it is.

The emptyCell function takes the board as input and returns the indices of the first empty cell it encounters in the board array. It returns [-1, -1, 0] if there are no empty cells.

The function sudokuSolving receives the board and a number as inputs and returns True if the board has a singular solution that does not include the given number. It uses a backtracking technique to iteratively attempt various integers in vacant cells until it comes up with a workable answer. It returns False if the requested number is discovered throughout this operation.

Future directives for this code might call for enhancing the game's user interface to make it more engaging and visually appealing. The code might also be changed to provide scoring and multiplayer games. To shorten the runtime for creating puzzles and solving boards, the code may also be optimized. The code might also be more adaptable if the user could input their own unique problems to solve. As soon as the students mastered a more advanced level of Python programming, it could be utilized.

Overall, the code is a well-structured implementation of the Sudoku game with all the necessary functions to generate, display, and solve the puzzle. The code also allows for a daily challenge and different difficulty levels.

VII. References

5 Real World Applications Of Sudoku. (2020, May 25). Sudoku Lovers - Play Online Sudoku for Free!
https://www.sudokulovers.com/5-real-world-applications-of-sudoku

Fischer, K. (2019, May 22). Sudoku or Crosswords May Help Keep Your Brain 10 Years Younger. Healthline; Healthline Media.
https://www.healthline.com/health-news/can-sudoku-actually-keep-your-mind-sharp

Li, S., Cui, J., & Gao, J. (2015). Effect of Sudoku game on cognitive function in elderly. Modern Hospital, 15(5), 52-54.

Pozza, R. D., Bravi, L., & Buono, A. (2021). Development of a Sudoku game in Python. International Journal of Computer Science Issues, 18(1), 73-80.

The History of Sudoku | Play Free Sudoku, a Popular Online Puzzle Game. (n.d.). Sudoku.com. https://sudoku.com/how-to-play/the-history-of-sudoku/#:~:text=The%20game%20first%20appeared%20in

The Math Behind Sudoku: Solving Strategy. (n.d.). Pi.math.cornell.edu. Retrieved April 15, 2023, from http://pi.math.cornell.edu/~mec/Summer2009/Mahmood/Solve.html#:~:text=The%20most%20basic%20strategy%20to

Uniyal, P. (2022, September 9). International Sudoku Day 2022: Health benefits of playing Sudoku every day. Hindustan Times. https://www.hindustantimes.com/lifestyle/health/international-sudoku-day-2022-health-benefits-of-playing-sudoku-every-day-101662724721233.html

VIII. Appendices

A. User's Manual

SudokuPro is a Python-based program that allows users to solve Sudoku puzzles. It provides an interactive interface for users to input their Sudoku board and then solves it using a backtracking algorithm. Additionally, the program keeps track of the time taken to solve the puzzle and stores it on a leaderboard, allowing users to compare their solving times with others.

The Sudoku Solver program offers the following features:

1. Interactive Sudoku board input: Users can input their Sudoku board using a simple text-based interface. The program displays the original board and allows users to enter their guesses for the empty cells.
2. Sudoku board validation: The program checks for invalid inputs and ensures that the entered board is valid with proper constraints.

3. Sudoku puzzle solving: The program uses a backtracking algorithm to solve the input Sudoku puzzle. It provides a solution to the puzzle if it is solvable, or notifies the user if the puzzle is unsolvable.
4. Solution display: The program displays the solved Sudoku board after solving the puzzle. Users can compare their solution with the original board to verify correctness.
5. Time tracking: The program records the time taken to solve the puzzle and displays it to the user.

To use the Sudoku Solver program, follow these steps:

1. Run the Program: Run the Sudoku Solver program in your Python environment.
2. Input Sudoku Board: The program will display an example Sudoku board. You can replace the numbers in the board with your own puzzle. Enter the numbers row by row, separating them with spaces, and use '0' to represent empty cells.
3. Solve the Puzzle: After inputting the Sudoku board, the program will start solving the puzzle using a backtracking algorithm. It will display the solved Sudoku board once the puzzle is solved, or notify you if the puzzle is unsolvable.
4. Record Your Time: The program will display the finishing time once the puzzle is solved. Note down your time for future reference.

The Sudoku Solver program provides an interactive way to input, solve, and track your Sudoku puzzle-solving times. It is a useful tool for Sudoku enthusiasts who want to practice their skills and compete with others. Enjoy using the Sudoku Solver program!

B. Poster

C. Source Code

```
import datetime

import numpy as np

import random

import time


def generateBoard(board):
```

```
    for i in range(9):
        for j in range(9):
            if j==0:
                print("|",end='')
            if j!=8:
                print(board[i,j],end=' ')
            else:
                print(board[i,j],end='')
            if (j+1)%3==0:
                print("|",end='')
        if (i+1)%3==0:
            print("\n----------------------",end='')
        print()

def showSolvedBoard(board):
    print("Solved Board:")
    generateBoard(board)

def emptyCell(board):
    for i in range(9):
        for j in range(9):
            if board[i,j]==0:
                row=i
                col=j
                Fill_Chk=1
                res=np.array([row,col,Fill_Chk],dtype="int8")
                return res
    res=np.array([-1,-1,0])
    return res

def isValid(board,row,col,num):
```

```
    row_start=(row//3)*3
    col_start=(col//3)*3
    if num in board[:,col] or num in board[row,:]:
        return False
    if num in board[row_start:row_start+3,col_start:col_start+3]:
        return False
    return True

def generatePuzzle(board,difficulty):
    count,done=0,False
    if difficulty == "Easy":
        print("Easy Difficulty Puzzle Generating...\n\n")
        upper_limit=2
    elif difficulty == "Medium":
        print("Medium Difficulty Puzzle Generating...\n\n")
        upper_limit=35
    else:
        print("Hard Difficulty Puzzle Generating...\n\n")
        upper_limit=41
    while True:
        i=random.randint(0,8)
        j=random.randint(0,8)
        if count<=upper_limit:
            if board[i,j]!=0:
                not_check=board[i,j]
                board[i,j]=0
                board_copy=board
                if sudokuSolving(board_copy,not_check):
                    board[i,j]=not_check
                    continue
                row_start=(i//3)*3
```

```
            col_start=(j//3)*3
            if difficulty == "Easy":
                if
np.count_nonzero(board[row_start:row_start+3,col_start:col_start+3])<5:
                    board[i,j]=not_check
                    continue
            elif difficulty == "Medium":
                if
np.count_nonzero(board[row_start:row_start+3,col_start:col_start+3])<4:
                    board[i,j]=not_check
                    continue
            else:
                if
np.count_nonzero(board[row_start:row_start+3,col_start:col_start+3])<3:
                    board[i,j]=not_check
                    continue
            count+=1
        else:
            done=True
            break

def Play_Sudoku(solvedBoard,unsolvedBoard):
    while True:
        row=int(input("Enter the row to insert number:")) - 1
        col=int(input("Enter the column to insert number:")) - 1
        number_check=int(input("Enter the number(or press 10 to exit):"))
        if number_check!=10:
            if unsolvedBoard[row,col]==0:
                print(solvedBoard[row,col])
                if solvedBoard[row,col]==number_check:
                    print("Correct! Updated board:")
```

```
                unsolvedBoard[row,col]=number_check
                generateBoard(unsolvedBoard)
            else:
                print("Incorrect!Updated board:")
                generateBoard(unsolvedBoard)
        else:
            print("That location is already correctly filled!")
        if np.array_equal(solvedBoard,unsolvedBoard):
            print("Congrats on solving the sudoku!")
            break
    else:
        print("\nThe solved board is:")
        generateBoard(solvedBoard)
        print("\nThank you for playing!\nWe hope to see you
again.\nRegards,\nYour friendly neighbourhood programmer")
        return

def sudokuSolving(board,not_check):
    x=emptyCell(board)
    if x[2]==0:
        return True
    else:
        row=x[0]
        col=x[1]
        for i in np.random.permutation(10):
            if i!=0 and i!=not_check:
                if isValid(board,row,col,i):
                    board[row,col]=i
                    if sudokuSolving(board,not_check):
                        return True
                    board[row,col]=0
```

```
    return False


def main():
    ch = int(input("Hello! Choose the level of difficulty:\n1. Easy\n2. Medium\n3.
Hard\n4. Daily Challenge\nYour choice:"))
    if ch == 1:
        difficulty = "Easy"
    elif ch == 2:
        difficulty = "Medium"
    elif ch == 3:
        difficulty = "Hard"
    elif ch == 4:
        today = datetime.date.today()
        challenge_seed = today.toordinal()
        random.seed(challenge_seed)
        difficulty = "Daily Challenge"
    else:
        print("Invalid choice. Exiting...")
        return
    start_time = time.time()
    board = np.zeros((9, 9), dtype="int8")
    if sudokuSolving(board, -1):
        solvedBoard = board.copy()
        print("\n\nThe unsolved puzzle is:\n")
        generatePuzzle(board, difficulty)
        generateBoard(board)
        unsolvedBoard = board.copy()
        Play_Sudoku(solvedBoard, unsolvedBoard)
    else:
        print("The board is not possible!")
    end_time = time.time()
```

```
    elapsed_time = end_time - start_time
    print("Finished Time: {:.2f} seconds".format(elapsed_time))

if __name__ == "__main__":
    main()
```

D. Work Breakdown

Student Name	Tasks Assigned	Percentage of the Work Contribution
CRUZ	- Source Code - User's Manual - Review of Related Literature	40%
ITCHON	- Hierarchy Chart - Flowchart - Pseudocode - Results - Discussion of Results - Analysis, Conclusion, and Future Directives	30%
YANO	- Introduction - Background of the Study - Problem Statement - Objectives - Significance of the Project - Game Poster	30%

E. Personal Data Sheet

CRUZ, Jose Andres A.

ITCHON, Patrice Tracy P.

YANO, Alexis Mae O.