Mastermind

T1 Planning and Design based on Mathematical and Declarative Programming techniques:

MasterMind is a code-breaking game that requires players to guess a secret code composed of colored pegs. The game is designed to test logical reasoning and deduction skills. This document outlines the planning and design of the MasterMind game using mathematical and declarative programming techniques.

Mathematical Structures:

Following mathematical structures were used:

- **Combinatorics:** The game leverages combinatorial principles to generate and evaluate guesses. The generate_code function creates combinations of colors, and the evaluate_guess function checks for permutations and combinations of these colors to provide feedback on the guess.
- **Probability:** The random generation of the secret code ensures that each game is unique and unpredictable, which adds to the challenge. The uniform distribution of color choices ensures fairness.
- **Set Theory:** Evaluating guesses involves comparing sets of colors. The evaluate_guess function uses set operations to determine how many colors in the guess match the secret code, both in the correct and incorrect positions.

Declarative Programming Techniques:

Following declarative programming techniques were used:

- **List Comprehensions:** The 'generate_code' function uses a list comprehension to create the secret code in a concise and readable manner:
- **Functional Programming:** The 'evaluate_guess' function applies functional programming techniques by using generator expressions to calculate 'correct_pos' and 'total_correct':

- **Recursion and Iteration:** The game loop and player input handling are designed using iterative constructs to repeatedly prompt the user and evaluate their input until the game ends.

Software Engineering Principles

In the making of this game, following software engineering principles were used:

- **Modularity:** The game is structured into distinct functions ('generate_code', 'human_guess_code', 'evaluate_guess', etc.), each responsible for a specific task. This makes the code easier to understand, test, and maintain.
- **Abstraction:** High-level functions like 'play_human_vs_human', 'play_human_vs_cpu', and 'play_campaign' abstract the game's core logic, making the code more readable and reducing complexity.
- **Encapsulation:** Data and functions are encapsulated within the game logic. For example, the secret code generation and evaluation are handled within their respective functions, protecting the internal state from unintended modifications.
- **Separation of Concerns:** The game logic is separated from user input and output. Functions that handle game mechanics (like 'evaluate_guess') are distinct from those that manage interaction with the player (like 'human_guess_code').

The MasterMind game implementation demonstrates the effective use of software engineering principles, mathematical techniques, and declarative programming. By carefully planning and designing the game, we ensure a maintainable, and enjoyable experience for players. This approach not only enhances the quality of the code but also provides a strong foundation for future enhancements and scalability.

T2 Program Implementation based on Declarative Programming tools and techniques.

Source Code:

```
'''

MasterMind Game:
Programming Language: Python
Interface: CLI based
Student Name: Aisha Abdi
```

```
Student ID: 22160571

'''

import random

COLORS = ["R", "G", "B", "Y", "W", "O"]
TRIES = 10
CODE_LENGTH = 4

def generate_code():
    """Generate a random secret code."""
    code = [random.choice(COLORS) for _ in range(CODE_LENGTH)]
    return code

def human_guess_code():
    """Allow a human player to guess the code."""
    while True:
        guess = input("Enter your guess (space-separated colors):
").upper().split()
        if len(guess) != CODE_LENGTH:
            print(f"You must guess {CODE_LENGTH} colors.")
            continue
        if all(color in COLORS for color in guess):
            break
        else:
            print("Invalid colors. Please try again.")
    return guess

def evaluate_guess(guess, code):
    """Evaluate the guess and return the number of correct and incorrect
positions."""
    correct_pos = sum(1 for g, c in zip(guess, code) if g == c)
    total_correct = sum((guess.count(color) if guess.count(color) <=
code.count(color) else code.count(color)) for color in COLORS)
    incorrect_pos = total_correct - correct_pos
    return correct_pos, incorrect_pos

def play_human_vs_human():
    """Start a new 2-player game."""
    print("Player 1, create a secret code.")
    code = generate_code()
    print("Player 2, try to guess the code!")
    for attempt in range(1, TRIES + 1):
        print(f"Attempt {attempt}:")
        guess = human_guess_code()
        correct_pos, incorrect_pos = evaluate_guess(guess, code)
        if correct_pos == CODE_LENGTH:
```

```
            print("Congratulations! Player 2 guessed the code!")
            break
        print(f"Correct Positions: {correct_pos} | Incorrect Positions:
{incorrect_pos}")
    else:
        print("Player 2 ran out of tries. Player 1 wins!")
    print(f"The secret code was: {' '.join(code)}")

def play_human_vs_cpu():
    """Start a new single-player game against the CPU."""
    print("The CPU will create a secret code.")
    code = generate_code()
    print("Try to guess the code!")
    for attempt in range(1, TRIES + 1):
        print(f"Attempt {attempt}:")
        guess = human_guess_code()
        correct_pos, incorrect_pos = evaluate_guess(guess, code)
        if correct_pos == CODE_LENGTH:
            print("Congratulations! You guessed the code!")
            break
        print(f"Correct Positions: {correct_pos} | Incorrect Positions:
{incorrect_pos}")
    else:
        print("You ran out of tries. The CPU wins!")
    print(f"The secret code was: {' '.join(code)}")

def play_campaign():
    """Start a new single-player 'campaign' against the CPU."""
    # Define multiple levels with predefined codes
    levels = [
        ['R', 'G', 'B', 'Y'],  # Level 1
        ['R', 'R', 'G', 'G'],  # Level 2
        ['B', 'B', 'B', 'W']   # Level 3
    ]

    print("Welcome to the campaign mode!")
    for level, code in enumerate(levels, 1):
        print(f"Level {level}:")
        print("The CPU has created a secret code.")
        print("Try to guess the code!")
        for attempt in range(1, TRIES + 1):
            print(f"Attempt {attempt}:")
            guess = human_guess_code()
            correct_pos, incorrect_pos = evaluate_guess(guess, code)
            if correct_pos == CODE_LENGTH:
                print("Congratulations! You guessed the code!")
                break
```

```
            print(f"Correct Positions: {correct_pos} | Incorrect Positions:
{incorrect_pos}")
        else:
            print("You ran out of tries. The CPU wins!")
        print(f"The secret code was: {' '.join(code)}")
        if level < len(levels):
            next_level = input("Press Enter to proceed to the next level.")

def main():
    print("Welcome to Mastermind!")
    print("Choose a game mode:")
    print("1. Human vs Human")
    print("2. Human vs CPU")
    print("3. Campaign")

    while True:
        choice = input("Enter the number of your choice (1-3): ")
        if choice == '1':
            play_human_vs_human()
            break
        elif choice == '2':
            play_human_vs_cpu()
            break
        elif choice == '3':
            play_campaign()
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 3.")

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

Output Screenshots:

```
Welcome to Mastermind!
Choose a game mode:
1. Human vs Human
2. Human vs CPU
3. Campaign
Enter the number of your choice (1-3): 1
Player 1, create a secret code.
Player 2, try to guess the code!
Attempt 1:
Enter your guess (space-separated colors): R G B Y
Correct Positions: 0 | Incorrect Positions: 1
Attempt 2:
Enter your guess (space-separated colors): W W G B
Correct Positions: 0 | Incorrect Positions: 0
Attempt 3:
Enter your guess (space-separated colors): R R R R
Correct Positions: 3 | Incorrect Positions: 0
Attempt 4:
Enter your guess (space-separated colors): W R G B
Correct Positions: 1 | Incorrect Positions: 0
Attempt 5:
Enter your guess (space-separated colors): B R G B
Correct Positions: 1 | Incorrect Positions: 0
Attempt 6:
Enter your guess (space-separated colors): B R R R
Correct Positions: 3 | Incorrect Positions: 0
Attempt 7:
Enter your guess (space-separated colors): B G R W
Correct Positions: 1 | Incorrect Positions: 0
Attempt 8:
Enter your guess (space-separated colors): R B G B
Correct Positions: 0 | Incorrect Positions: 1
Attempt 9:
Enter your guess (space-separated colors): R G G B
Correct Positions: 0 | Incorrect Positions: 1
Attempt 10:
Enter your guess (space-separated colors): Y W R G
Correct Positions: 1 | Incorrect Positions: 0
Player 2 ran out of tries. Player 1 wins!
The secret code was: O R R R
```

```
Welcome to Mastermind!
Choose a game mode:
1. Human vs Human
2. Human vs CPU
3. Campaign
Enter the number of your choice (1-3): 2
The CPU will create a secret code.
Try to guess the code!
Attempt 1:
Enter your guess (space-separated colors): R G B Y
Correct Positions: 0 | Incorrect Positions: 2
Attempt 2:
Enter your guess (space-separated colors): R R G B
Correct Positions: 1 | Incorrect Positions: 2
Attempt 3:
Enter your guess (space-separated colors): B G R W
Correct Positions: 1 | Incorrect Positions: 1
Attempt 4:
Enter your guess (space-separated colors): W W R G
Correct Positions: 1 | Incorrect Positions: 1
Attempt 5:
Enter your guess (space-separated colors): W R R G
Correct Positions: 2 | Incorrect Positions: 1
Attempt 6:
Enter your guess (space-separated colors): B G R G
Correct Positions: 1 | Incorrect Positions: 1
Attempt 7:
Enter your guess (space-separated colors): B G R Y
Correct Positions: 1 | Incorrect Positions: 1
Attempt 8:
Enter your guess (space-separated colors): Y Y Y B
Correct Positions: 0 | Incorrect Positions: 0
Attempt 9:
Enter your guess (space-separated colors): Y R G W
Correct Positions: 1 | Incorrect Positions: 1
Attempt 10:
Enter your guess (space-separated colors): R G B Y
Correct Positions: 0 | Incorrect Positions: 2
You ran out of tries. The CPU wins!
The secret code was: G R R O
```

```
Welcome to Mastermind!
Choose a game mode:
1. Human vs Human
2. Human vs CPU
3. Campaign
Enter the number of your choice (1-3): 3
Welcome to the campaign mode!
Level 1:
The CPU has created a secret code.
Try to guess the code!
Attempt 1:
Enter your guess (space-separated colors): R G B Y
Congratulations! You guessed the code!
The secret code was: R G B Y
Press Enter to proceed to the next level.
Level 2:
The CPU has created a secret code.
Try to guess the code!
Attempt 1:
Enter your guess (space-separated colors): R B G B
Correct Positions: 2 | Incorrect Positions: 0
Attempt 2:
Enter your guess (space-separated colors): R R G G
Congratulations! You guessed the code!
The secret code was: R R G G
Press Enter to proceed to the next level.
Level 3:
The CPU has created a secret code.
Try to guess the code!
Attempt 1:
Enter your guess (space-separated colors): R B G Y
Correct Positions: 1 | Incorrect Positions: 0
Attempt 2:
Enter your guess (space-separated colors): B B B W
Congratulations! You guessed the code!
The secret code was: B B B W
```

Declarative Implementation of Mastermind in Python:

Data Structures and Type Definitions:

To implement the Mastermind game, we define a set of functions and constants that encapsulate the game logic and player interactions.

- **Constants:**

✓ COLORS: List of possible colors for the code.

✓ TRIES: Number of attempts a player has to guess the code.

✓ CODE_LENGTH: Length of the secret code.

- **Functions:**

✓ generate_code(): Generates a random secret code.

✓ human_guess_code(): Prompts the player to guess the code.

✓ evaluate_guess(guess, code): Evaluates the player's guess and returns the number of correct and incorrect positions.

✓ play_human_vs_human(): Manages the gameplay for human vs. human mode.

✓ play_human_vs_cpu(): Manages the gameplay for human vs. CPU mode.

✓ play_campaign(): Manages the gameplay for campaign mode.

✓ main(): Main function to start the game and handle game mode selection.

Behaviour Step Implementations:

- **Generating the Secret Code:** The generate_code function randomly selects colors to create a secret code.
- **Human Player Guess:** The human_guess_code function prompts the player to enter their guess, ensuring it meets the required format.
- **Evaluating the Guess:** The evaluate_guess function compares the player's guess to the secret code and returns the number of correct positions and incorrect positions.
- **Human vs. Human Gameplay:** The play_human_vs_human function manages the gameplay for a two-player game where one player creates the code and the other tries to guess it.
- **Human vs. CPU Gameplay:** The play_human_vs_cpu function manages the gameplay for a single-player game where the CPU creates the secret code and the player tries to guess it.
- **Campaign Mode Gameplay:** The play_campaign function manages the gameplay for a single-player campaign mode with predefined levels.
- **Main Function:** The main function starts the game and prompts the player to choose a game mode.

Areas of Improvement:

- Allow the first player to manually set the secret code in human vs. human mode.
- Introduce a hint system to assist players struggling to guess the code.
- Improve the user interface by adding color-coded output to enhance readability.
- Provide clearer instructions and feedback for invalid inputs.

T3 Testing and Verification of Programs via appropriate tools and techniques

---

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TC1	Verify secret code generation	1. Start the game. 2. Check if a random secret code is generated.	None	A random code of length 4 consisting of valid colors is generated.	As Expected	PASS
TC2	Verify human player guess input	1. Start the game. 2. Enter a guess with valid colors and correct length.	Valid guess input	The guess is accepted and processed by the game.	As Expected	PASS
TC3	Verify evaluation of player guess	1. Start the game with a known secret code. 2. Make a guess. 3. Check the evaluation.	Secret code, Player guess	The evaluation shows the correct number of correct and incorrect positions.	As Expected	PASS
TC4	Verify game termination upon correct guess	1. Start the game with a known secret code.	Secret code, Correct guess	The game terminates and congratulates the player.	As Expected	PASS

		2. Make a correct guess.				
TC5	Verify game termination upon running out of tries	1. Start the game. 2. Exhaust all tries without guessing the correct code.	None	The game terminates and declares the CPU or player as the winner.	As Expected	PASS