Mastermind Solver Project

BY: BASHIRAT SHAIBU
UMAR FARUK ABUBAKAR
VICTOR KEVIN
MALEEK ISA

INTRODUCTION

Mastermind is a classic code-breaking game that requires logic and reasoning. In this project, we implemented an AI-based solver for the game using a logical elimination strategy. The goal is to demonstrate how Artificial Intelligence can be used to solve constraint-based problems using search and optimization techniques.

PROBLEM DESCRIPTION

In the Mastermind game, a secret code consisting of four digits is generated, where each digit can range from 0 to 5. The challenge is to guess the code in as few attempts as possible. After each guess, feedback is provided in terms of how many digits are correct and in the correct position (black pegs) and how many digits are correct but in the wrong position (white pegs). This creates a logical problem where each guess must reduce the number of possible remaining codes.

This problem is considered a logic and optimization problem because the player (or algorithm) must use the feedback to eliminate impossible code combinations and narrow down the correct code. Artificial Intelligence techniques help automate this reasoning process effectively.

ALGORITHM USED

To solve the Mastermind game, we used a logical elimination strategy. First, we generate all possible code combinations. After making an initial guess, we receive feedback in the form of black and white pegs. We then eliminate any combination from the list of possibilities that would not produce the same feedback if it were the secret code. This reduces the search space with each guess.

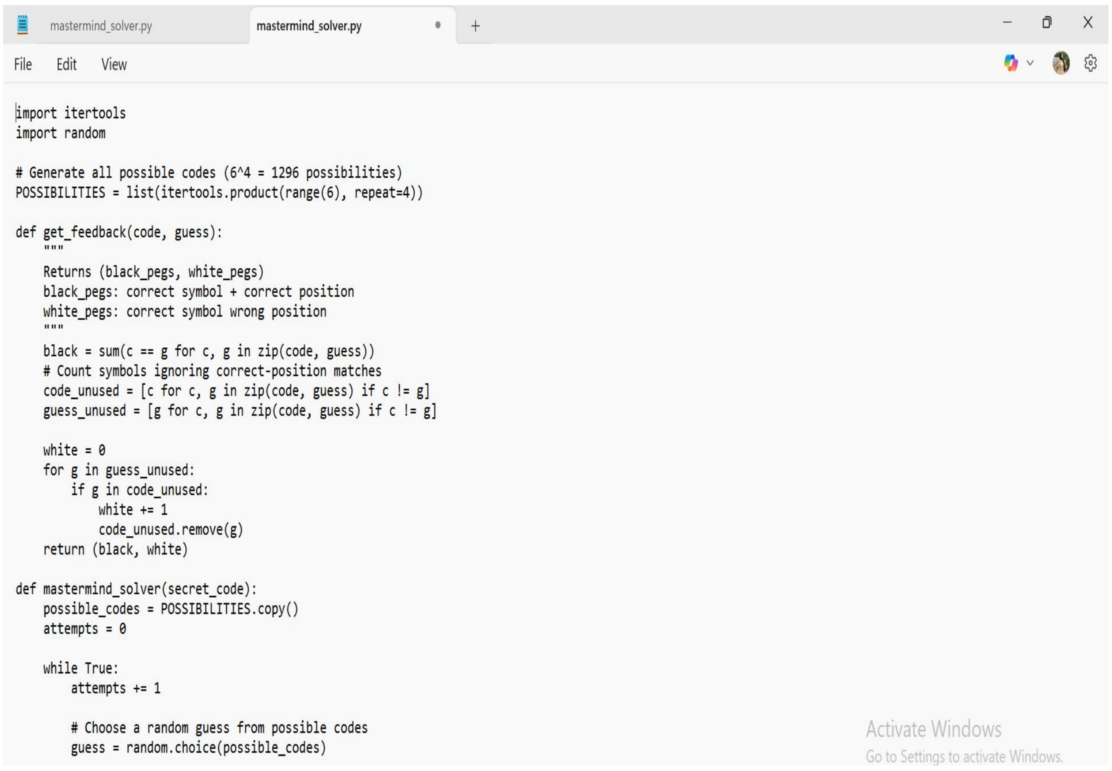
The process continues until the feedback indicates that all four digits are correct and in the right position (4 black pegs). This method is simple but efficient and falls under problem-solving and search algorithms in Artificial Intelligence.

IMPLEMENTATION(CODE)

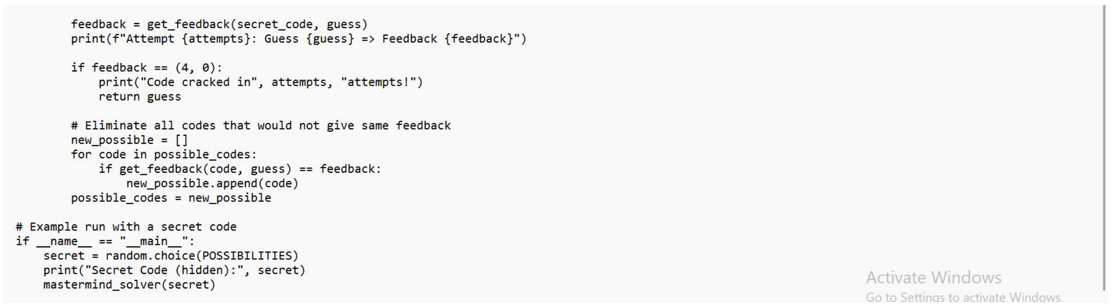
Below is the Python implementation of the Mastermind solver using logical elimination.

```
import itertools
import random

# Generate all possible codes (6^4 = 1296 possibilities)
```

```
POSSIBILITIES = list(itertools.product(range(6), repeat=4))

def get_feedback(code, guess):
    black = sum(c == g for c, g in zip(code, guess))
    code_unused = [c for c, g in zip(code, guess) if c != g]
    guess_unused = [g for c, g in zip(code, guess) if c != g]
    white = 0
    for g in guess_unused:
        if g in code_unused:
            white += 1
            code_unused.remove(g)
    return (black, white)

def mastermind_solver(secret_code):
    possible_codes = POSSIBILITIES.copy()
    attempts = 0
    while True:
        attempts += 1
        guess = random.choice(possible_codes)
        feedback = get_feedback(secret_code, guess)
        print(f"Attempt {attempts}: Guess {guess} => Feedback {feedback}")
        if feedback == (4, 0):
            print("Code cracked in", attempts, "attempts!")
            return guess
        new_possible = []
        for code in possible_codes:
            if get_feedback(code, guess) == feedback:
                new_possible.append(code)
        possible_codes = new_possible

if __name__ == "__main__":
    secret = random.choice(POSSIBILITIES)
    print("Secret Code (hidden):", secret)
    mastermind_solver(secret)
```

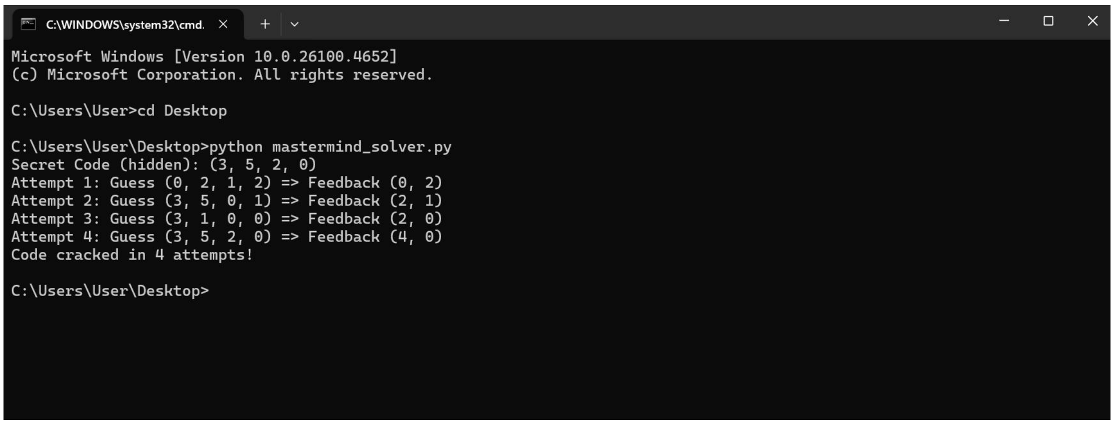


```
import itertools
import random

# Generate all possible codes (6^4 = 1296 possibilities)
POSSIBILITIES = list(itertools.product(range(6), repeat=4))

def get_feedback(code, guess):
    """
    Returns (black_pegs, white_pegs)
    black_pegs: correct symbol + correct position
    white_pegs: correct symbol wrong position
    """
    black = sum(c == g for c, g in zip(code, guess))
    # Count symbols ignoring correct-position matches
    code_unused = [c for c, g in zip(code, guess) if c != g]
    guess_unused = [g for c, g in zip(code, guess) if c != g]

    white = 0
    for g in guess_unused:
        if g in code_unused:
            white += 1
            code_unused.remove(g)
    return (black, white)

def mastermind_solver(secret_code):
    possible_codes = POSSIBILITIES.copy()
    attempts = 0

    while True:
        attempts += 1

        # Choose a random guess from possible codes
        guess = random.choice(possible_codes)
```



```
        feedback = get_feedback(secret_code, guess)
        print(f"Attempt {attempts}: Guess {guess} => Feedback {feedback}")

        if feedback == (4, 0):
            print("Code cracked in", attempts, "attempts!")
            return guess

        # Eliminate all codes that would not give same feedback
        new_possible = []
        for code in possible_codes:
            if get_feedback(code, guess) == feedback:
                new_possible.append(code)
        possible_codes = new_possible

# Example run with a secret code
if __name__ == "__main__":
    secret = random.choice(POSSIBILITIES)
    print("Secret Code (hidden):", secret)
    mastermind_solver(secret)
```

OUTPUT SCREENSHOT



```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>cd Desktop

C:\Users\User\Desktop>python mastermind_solver.py
Secret Code (hidden): (3, 5, 2, 0)
Attempt 1: Guess (0, 2, 1, 2) => Feedback (0, 2)
Attempt 2: Guess (3, 5, 0, 1) => Feedback (2, 1)
Attempt 3: Guess (3, 1, 0, 0) => Feedback (2, 0)
Attempt 4: Guess (3, 5, 2, 0) => Feedback (4, 0)
Code cracked in 4 attempts!

C:\Users\User\Desktop>
```

CONCUSION

This project demonstrates how a logical elimination algorithm can be used to solve the Mastermind game efficiently. By applying feedback from each guess to reduce the search space, our AI program successfully identified the correct code within a few attempts. This shows the power of Artificial Intelligence in solving logic-based problems and optimizing information through reasoning.