# MEMORANDUM

**10/9/23**

**TO:** Charlie Revfem, Lecturer, Mechanical Engineering
**FROM:** Noah Tanner – ME 405
**COPIES:** None
**SUBJECT:** **HW0x00 - Mastermind**

## Introduction

In this homework assignment, I used a finite-state machine to represent the game of Mastermind, and then implemented the game through python and interaction with the command line. The game follows a similar flow to world, where a user interacts with the values 0, 1, 2, 3, 4, and 5, creating 4-number sequences and trying to guess the correct code. After each entry, in a Wordle style, the game will evaluate the guesses by showing which numbers are in the correct index with a "+", and which guesses are correct, but in the wrong index with a "-". In alignment with the requirements, the code must allow for repetitive play, keeping track of variables such as win and loss count, as well as employ proper error handling for invalid guesses and other errors that a user might unknowingly create.

## Software Setup

The program runs in one file, called tanner_main.py. Figure 1 illustrates the finite state machine that was created and implemented in the code. There are 6 states total. The main logic that determines the validity of the guesses is implemented within State 3 and State 4. In State 3, the entered guess is compared directly against the correct code by index. Direct matches are recorded and changed to an 'X' so that they will not interfere in State 4. In State 4, each non 'X' value in the entered guess is compared against each non 'X' value in the correct code to determine if there are any values that are the correct number, but the wrong index. Throughout each of these states, a correct value will add a '+' to a string variable and a misplaced value will add a '-' to a string variable that will be printed at each guess, indicating the correctness of the current guess.

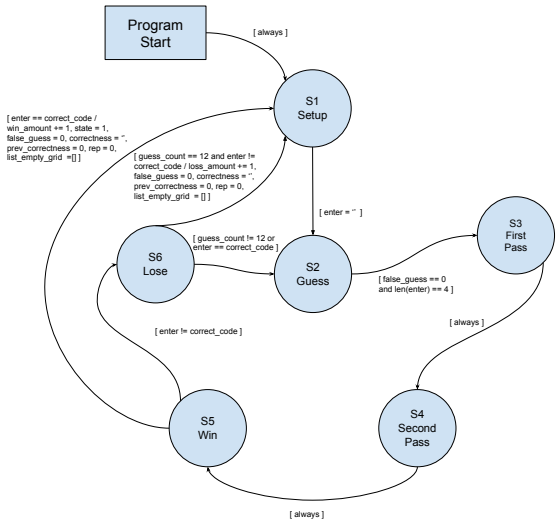


*Figure 1: Mastermind Finite State Machine*

**Conclusion**

Having previously implemented finite state machines in assembly throughout ME 305, doing the same in Python was significantly easier. This assignment provided a valuable and challenging logic puzzle that quickly familiarized me with writing python programs and debugging them. A tool that I used consistently throughout the programming process was break-points in VScode as well as the Debugging Window. The breakpoints allowed me to stop at certain states and the Debugging Window would allow me to evaluate expressions and variables to determine where I was having errors in upcoming lines. Furthermore, this project introduced me to various data types and how to mutate them. Stress testing the project to ensure that there it was “idiot-proof” was an enjoyable moment as well, as I got to watch my friends try and beat my program.

## Appendix

### A. Source Code

```
"""

HW1 - Mastermind Game
Written by Noah Tanner
ME 405 - Professor Refvem
Fall 2023

"""

# imports
import random
import sys

# definitions and initialization
# set state into for state automatically
state = 1
win_amount = 0
loss_amount = 0
false_guess = 0
correctness = ''
prev_correctness = 0
rep = 0
first_run = 0
spoiled_mode = 0

# print statements
empty_grid = """
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
```

```
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    |   |   |   |   |
    +---+---+---+---+
    """
list_empty_grid = []
for let in empty_grid:
    list_empty_grid.append(let)
wins = "wins: "
losses = "losses: "
start_text = """                    Welcome to the game of Mastermind!

            You will be the codebreaker. Try to break the secret code by entering 4-
digit
            codes using the numbers 0-5. Your guesses will be marked with a (+) symbol
for
            each correct value in the correct location and a (-) for each correct
value in
            an incorrect location.

                                        Press enter to begin
            """
new_game = "Mastermind! Try to break the code. \n"


# grid layout for the beginning of the game

while True:
    if state == 1:
        # state 1 - setup
        # print starting text
        if first_run == 0:
            print(start_text)
            while True:
                enter = input('')
                if not enter:
                    break
            first_run += 1
```

```
        state = 2                                                   # set next state
        guess_count = 0                                             # initialize the
guess count to 0 guesses, used for printing board

        correct_code = ''.join(random.choices('012345', k = 4))     # generate the
random code to be guessed

        if first_run == 1:
            first_run += 1
            print("Press 1 to play game in spoiled mode, otherwise press any key...")
            while True:
                spoiled = input('')
                if spoiled == '1' :
                    spoiled_mode = 1
                    break
                else:
                    break

        # new game prints
        print(new_game)
        print(wins + ' ' + str(win_amount))
        print(losses + ' ' + str(loss_amount))
        print(empty_grid)


    # state 2 - guess
    if state == 2:

        while True:
            enter = input('Enter a guess: ')
            for n in enter:
                if n not in ['0', '1', '2', '3', '4', '5']:
                    false_guess += 1
                    True
            for n in enter:
                if n in ['0', '1', '2', '3', '4', '5']:
                    false_guess = 0
            if false_guess != 0 or len(enter) != 4:
                print('Invalid entry, try again')

            else:
                false_guess = 0
                break

        # start checking values now that we have a valid entry
```

```
        state = 3

    # state 3 - first pass
    if state == 3:
        list_enter = [ val for val in enter ]
        list_correct_code = [ val for val in correct_code ]
        for idx in range(4):
            if list_enter[idx] == correct_code[idx]:
                correctness = correctness + '+'
                list_enter[idx] = 'X'                                   # set the idxs
of temp lists to 'X' so they aren't reused later
                list_correct_code[idx] = 'X'
        state = 4

    # state 4 - second pass
    if state == 4:
        for i in range(4):
            for j in range(4):
                if list_enter[i] == list_correct_code[j] and list_enter[i] != 'X':
                    correctness = correctness + '-'
                    list_enter[idx] = 'X'                               # set the idxs
of temp lists to 'X' so they aren't reused later
                    list_correct_code[idx] = 'X'

        # print results from second pass
        for idx in range(0,4):
            if idx == 0:
                list_empty_grid[-42-(44*guess_count)-prev_correctness] = enter[0]
            if idx == 1:
                list_empty_grid[-38-(44*guess_count)-prev_correctness] = enter[1]
            if idx == 2:
                list_empty_grid[-34-(44*guess_count)-prev_correctness] = enter[2]
            if idx == 3:
                list_empty_grid[-30-(44*guess_count)-prev_correctness] = enter[3]

        # append the correctness count on the right side
        correctness_list = [str(let) for let in correctness]
        if guess_count == 0:
            index_to_insert = -27
        elif guess_count > 0:
            index_to_insert = -27 - (44* guess_count) - prev_correctness
        for val in correctness_list:
            list_empty_grid.insert(index_to_insert, val)
        #list_empty_grid.insert(index_to_insert, '\n')
        new_grid_print = ''.join(list_empty_grid)
```

```
        if spoiled_mode == 1:
            print('Code to break: ' + str(correct_code))
        print(new_grid_print)
        guess_count += 1                                        # increment guess
count after each print, first go needs to have 0!
        for val in correctness:
            prev_correctness += 1
        correctness = ''                                        # reset
correctness for next guess
        state = 5                                               # set next state

    # state 5 - win
    if state == 5:
        if enter == correct_code:
            win_amount += 1
            print('You win!\n')
            again = input('Play again? Enter (y) for yes: ')
            if again == ('y' or 'Y'):
                # reinitialize all the relevant state vars and reset state to 1
                state = 1
                false_guess = 0
                correctness = ''
                prev_correctness = 0
                correctness = ''
                rep = 0
                list_empty_grid = []
                for let in empty_grid:
                    list_empty_grid.append(let)
            else:
                print('Thanks for playing!\n')
                print(' Total wins: ' + str(win_amount) + '\n' )
                print(' Total losses: ' + str(loss_amount) + '\n' )
                while True:
                    exit_program = input('Press enter to exit... ')
                    if exit_program == '':
                        sys.exit()
                    else:
                        'Welcome to the backrooms...'
        else:
            state = 6

    # state 6 - lose
    if state == 6:
        if guess_count == 12 and enter != correct_code:
            loss_amount += 1
```

```
            # loss conditions
            print('You lose!\n')
            again = input('Play again? Enter (y) for yes: ')
            if again == ('y' or 'Y'):
                # reinitialize all the relevant state vars and reset state to 1
                state = 1
                false_guess = 0
                correctness = ''
                prev_correctness = 0
                correctness = ''
                rep = 0
                list_empty_grid = []
                for let in empty_grid:
                    list_empty_grid.append(let)
            else:
                print('Thanks for playing!\n')
                print(' Total wins: ' + str(win_amount) + '\n' )
                print(' Total losses: ' + str(loss_amount) + '\n' )
                while True:
                    exit_program = input('Press enter to exit... ')
                    if exit_program == '':
                        sys.exit()
                    else:
                        'Welcome to the backrooms...'
            pass
        else:
            state = 2
```