CS5131
Introduction to Artificial Intelligence

Name: ________________________________ (   ) Date: _______________

Assignment 2 – 50 Marks

Mastermind is a code-breaking game for two players. It resembles an earlier pencil and paper game called Bulls and Cows that may date back a century. A computer adaptation of it was run in the 1960s on Cambridge University's Titan computer system, where it was called "MOO".

The modern game with pegs was invented in 1970 by Mordecai Meirowitz, an Israeli postmaster and telecommunications expert. He presented the idea to many major toy companies and the intellectual rights to the game was eventually purchased up by Invicta Plastics after it was exhibited at the Nuremberg International Toy Fair.

Presently, a commercial board game exists in two variants: "classic" Mastermind with four positions and six colors, and "super" Mastermind with five positions and eight colors. The classic game is played using a decoding board, with a shield at one end covering a row of four large holes, and 12 additional rows containing four large holes next to a set of four small holes; code pegs of six different colors (or more; see variations below), with round heads, which will be placed in the large holes on the board; and key pegs, some colored black, some white, which are flat-headed and smaller than the code pegs; they will be placed in the small holes on the board.

Game	Year	Colors	Holes
Mastermind	1972	6	4
Bagels[20]	1972	10 digits	3
Royale Mastermind	1972	5 colors × 5 shapes	3
Mastermind44	1972	6	5
Grand Mastermind	1974	5 colors × 5 shapes	4
Super Mastermind (a.k.a. *Deluxe Mastermind*; a.k.a. *Advanced Mastermind*)	1975	8	5
Word Mastermind	1975	26 letters	4
Mini Mastermind	1976	6	4
Number Mastermind	1976	6 digits	4
***Electronic Mastermind* (Invicta)**	**1977**	**10 digits**	**3, 4, or 5**
Super-Sonic Electronic Mastermind (Invicta)	1979	10 digits	3, 4, 5, or 6
Walt Disney Mastermind	1978	5	3
Mini Mastermind (a.k.a. *Travel Mastermind*)	1988	6	4
Mastermind Challenge	1993	8	5
Parker Mastermind	1993	8	4
Mastermind for Kids	1996	6	3
Mastermind Secret Search	1997	26 letters	3-6
Electronic Hand-Held Mastermind (Hasbro)	1997	6	4
New Mastermind	2004	8	4
Mini Mastermind	2004	6	4

Objectives:

- To outsmart your opponent with a clever code or great guesswork.
- As the Codemaker: your goal is to set a mystery code so cunning that it will keep your opponent guessing for as long as possible.
- As the Decoder: you must break the secret code in the fewest number of guesses.

Game Play and Rules

The classic game is played using:

- a *decoding board*, with a *shield* at one end covering a row of four large holes, and twelve additional rows containing four large holes next to a set of four small holes
- *code pegs* of six different colors with round heads, which will be placed in the large holes on the board
- *key pegs*, some colored black, some white, which are flat-headed and smaller than the code pegs; they will be placed in the small holes on the board.

One player becomes the *codemaker*, the other the *codebreaker*. The codemaker chooses a pattern of four code pegs. Players decide in advance whether duplicates and blanks are allowed. If so, the codemaker may even choose four same-colored code pegs or four blanks. If blanks are not allowed in the code, the codebreaker may not use blanks in their guesses. The codemaker places the chosen pattern in the four holes covered by the shield, visible to the codemaker but not to the codebreaker.

The codebreaker tries to guess the pattern, in both order and color, within eight to twelve turns. Each guess is made by placing a row of code pegs on the decoding board. Once placed, the codemaker provides feedback by placing from zero to four key pegs in the small holes of the row with the guess. **A colored or black key peg is placed for each code peg from the guess which is correct in both color and position. A white key peg indicates the existence of a correct color code peg placed in the wrong position.**

If there are duplicate colors in the guess, they cannot all be awarded a key peg unless they correspond to the same number of duplicate colors in the hidden code. For example, if the hidden code is red-red-blue-blue and the player guesses red-red-red-blue, the codemaker will award two colored key pegs for the two correct reds, nothing for the third red as there is not a third red in the code, and a colored key peg for the blue. No indication is given of the fact that the code also includes a second blue.

Once feedback is provided, another guess is made; guesses and feedback continue to alternate until either the codebreaker guesses correctly, or all rows on the decoding board are full.

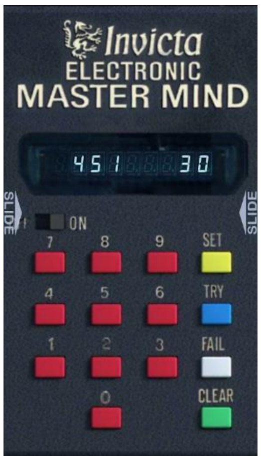
Mastermind Problem

Mastermind can be solved using various methods:

- Meta-heuristic
- Rule of thumb
- Logic programming
- Machine learning
- Genetic algorithm
- etc

Read **Effcient Solutions for Mastermind using Genetic Algorithms.pdf** provided.

For this assignment, all your programs, investigation, analysis, and evaluation should be carried out in Jupyter Notebook (**Assignment2_Your_Name.pynb**). Use markups to label the tasks, provide descriptions and your code should execute within the Jupyter Notebook. You are to perform the following tasks:



Task 1

Write a barebone program to simulate Electronic Mastermind 1977 in Python 3. The game rules are similar to the classic Mastermind except that ten digits will be used instead of colors and the player have a choice to play with 3, 4 or 5 codes.

Assumptions

- Entries are ordered and separated by a space.
- Blanks are not allowed.
- Duplicates are allowed.
- Key pegs are indicated by `'B'` (Correct digit and position) or `'W'` (Correct digit and wrong position). `'.'` indicates a wrong code.

Sample Output

```
ELECTRONIC MASTERMIND 1977
SELECT CODE TYPE: [3, 4, 5]: 3
[1, 0, 4] NOTE: THIS SHOULD BE HIDDEN. IT IS SHOWN FOR DEMO PURPOSES ONLY

Enter the code separated by space: 3 4 9
Guess 1 : 0 1 ['.', 'W', '.']

Enter the code separated by space: 1 4 0
Guess 2 : 1 2 ['B', 'W', 'W']

Enter the code separated by space: 1 1 1
Guess 3 : 1 0 ['B', '.', '.']

Enter the code separated by space: 1 9 4
Guess 4 : 2 0 ['B', '.', 'B']

Enter the code separated by space: 1 0 4
Guess 5 : 3 0 ['B', 'B', 'B']

SOLVED:  [1, 0, 4]

ELECTRONIC MASTERMIND 1977
SELECT CODE TYPE: [3, 4, 5]: 4
[3, 4, 8, 6] NOTE: THIS OUTPUT SHOULD BE HIDDEN. IT IS SHOWN FOR TESTING AND DEMO
PURPOSES ONLY

Enter the code separated by space: 3 3 3 3
Guess 1 : 1 0 ['B', '.', '.', '.']

Enter the code separated by space: 3 1 2 7
Guess 2 : 1 0 ['B', '.', '.', '.']

Enter the code separated by space: 3 6 4 8
Guess 3 : 1 3 ['B', 'W', 'W', 'W']

Enter the code separated by space: 3 8 4 6
Guess 4 : 2 2 ['B', 'W', 'W', 'B']

Enter the code separated by space: 3 4 8 6
Guess 5 : 4 0 ['B', 'B', 'B', 'B']

SOLVED:  [3, 4, 8, 6]
```

```
ELECTRONIC MASTERMIND 1977
SELECT CODE TYPE: [3, 4, 5]: 5
[7, 8, 2, 6, 7] NOTE: THIS OUTPUT SHOULD BE HIDDEN. IT IS SHOWN FOR TESTING AND DEMO
PURPOSES ONLY

Enter the code separated by space: 7 7 7 7 7
Guess 1 : 2 0 ['B', '.', '.', '.', 'B']

Enter the code separated by space: 7 1 2 3 7
Guess 2 : 3 0 ['B', '.', 'B', '.', 'B']

Enter the code separated by space: 7 4 2 5 7
Guess 3 : 3 0 ['B', '.', 'B', '.', 'B']

Enter the code separated by space: 7 6 2 8 7
Guess 4 : 3 2 ['B', 'W', 'B', 'W', 'B']

Enter the code separated by space: 7 8 2 6 7
Guess 5 : 5 0 ['B', 'B', 'B', 'B', 'B']

SOLVED:  [7, 8, 2, 6, 7]
```

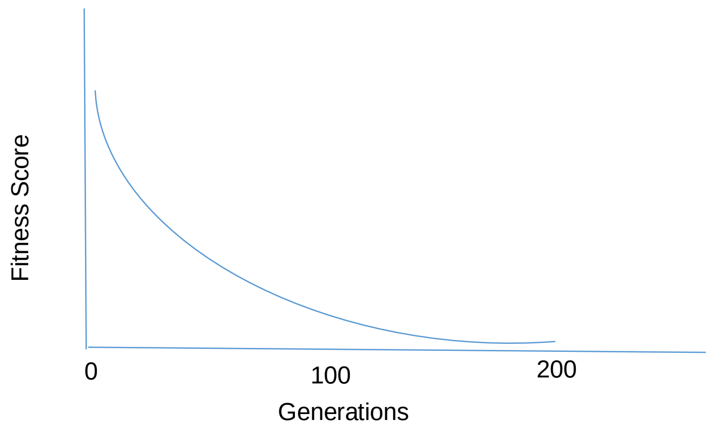
Task 2

Make a copy of the program in Task 1. Modify the program (Electronic Mastermind Solver) to solve Electronic Mastermind 1977 game automatically using **Genetic Algorithms (GA) ONLY**. Implement essential GA functions for fitness, selection, crossover, mutation, replacement, etc., to solve the secret code.

Ideally, you should seek to optimize your GA by experiencing with different crossover and mutation rates to find solutions with the least number of guesses. You can specify a maximum limit to the number of generations to run. For each generation, the following information will have to be displayed:

- Generation number
- Fitness score for a solution
- Any other information you deem is appropriate eg. population size, elapse time, cross over rate, mutation rate, elitism rate, intermediate guesses etc.

When your GA converged, the program should display the solved secret code. In addition, you should display a graph (matplotlib) to show how your GA converge to a solution over generations. Note: Your graph may look different depending on whether your fitness function is maximizing or minimizing.



Sample graph of fitness score after several generations

<u>Suggested Sample Output</u>

Note: You may choose to format your result differently to suit your style as long as the information on the progress is clear)

```
ELECTRONIC MASTERMIND 1977 GENETIC ALGORITHM SOLVER
Population   : xxx         Generation    : xxxx        Elitism: Yes
Mutation Rate: xxxx        Crossover Rate: xxxx        Elitism Rate: xxx

SELECT CODE TYPE: [3, 4, 5]: 3
[1, 0, 4] NOTE: THIS SHOULD BE HIDDEN. IT IS SHOWN FOR DEMO PURPOSES ONLY

GA: 3 4 9
Guess 1 : 0 1 ['.', 'W', '.']

GEN #: 1      Fitness: xxxxx   Elapse Time: xxxxxxx
…
GEN #: x      Fitness: xxxxx   Elapse Time: xxxxxxx


GA: 2 9 4

Guess 2 : 1 2 ['B', 'W', 'W']

GEN #: x      Fitness: xxxxx   Elapse Time: xxxxxxx
…
GEN #: x      Fitness: xxxxx   Elapse Time: xxxxxxx

GA: 1 1 1

Guess 3 : 1 0 ['B', '.', '.']

GEN #: x      Fitness: xxxxx   Elapse Time: xxxxxxx
…
GEN #: x      Fitness: xxxxx   Elapse Time: xxxxxxx

GA: 1 9 4

Guess 4 : 2 0 ['B', '.', 'B']

GEN #: x      Fitness: xxxxx   Elapse Time: xxxxxxx
…
GEN #: n      Fitness: xxxxx   Elapse Time: xxxxxxx

GA: 1 0 4

Guess 5 : 3 0 ['B', 'B', 'B']

SOLVED:  [1, 0, 4]
```

Task 3

Write a program (Electronic Mastermind Simulator) in Python 3 to simulate the problem-solving capabilities of your GA. The program should allow a user to specify the number simulations to run. For each run, a secret code should be generated and solved by your GA. Note: You do not need to display the gameplay for each run, however, you should dump the output of each secret code into a file (.csv or .txt) for data analysis, checking and verification purposes.

Run simulations to solve multiple secret codes (eg. 100 to 500). For each secret code display and keep track of the following:

- Total number of the generations taken to solve a secret code
- Elapse time to solve each secret code
- Number of guesses taken to solve the secret code
- Best fitness or Average Fitness scores
- Is the secret code solved within the predefined number of generations?

You should also consider dumping the above information to a file. Collect and compile information on the average number of generations taken, average time taken, success rate, best fitness score, type of selection, crossover and mutation rates used in each generation.

Investigate, analyse and evaluate your solutions. Attempt to answer the following questions:

- Do the performance of your solutions affected by different selection, crossover and mutation methods.
- What would be an ideal crossover and mutation rate?
- Does your GA performance degrades if more codes are use in the game?
- What is the average number of guesses needed to solve a 3, 4 or 5 code Mastermind game.
- Does using elitism help to improve the performance of your GA?

Explain and present your analysis with the aid of graph using Python data visualization libraries such as matplotlib, seaborn, etc.

Based on the research paper "**Effcient Solutions for Mastermind using Genetic Algorithms.pdf**" by Lotte Berghman et al, it was mentioned that GA's performance is comparable to that of other meta-heuristics for the standard setting with four positions and six colors, while it outperforms the existing algorithms when more colors and positions are considered. Comparing their conclusion with your own solutions, do you agree with the researcher's findings? Analyze and discuss.

Note:

- Program must be well commented. Poor programming style with ambiguous code will result in loss of marks,
- Explore using different mutation and crossover rate if your solution converged too slowly.
- You may want to set a limit to the number of generations to run for each simulation and monitor your GA's success rate.
- The best offspring in each generation will become the parent in the next generation.
- Assign the best offspring to become the parent of the next generation.
- Using the data generated in a round of simulation. Plot the graph of Best Individual Fitness against the Generation, using data visualization libraries.
- Save your program in as a Jupyter Notebook file (.pynb). (Note: Make sure your graph can be displayed in Jupyter Notebook before submitting)

Submission

- All relevant programs file must be zip.
- Name your submission as **Assignment2_Your_Name.zip**. Submission must be uploaded to Coursemology only. Email submission will be ignored and not mark.
- Work found to be plagiarized will be given zero mark.
- Deadline for submission: 17th **March 2022, Thursday, 2100**.
- A 2% penalty of the overall total mark will be imposed daily for late submission. Deduction will be capped at 50% of the assignment.

Grading Rubrics

- Functional Program
- Efficiency
- Correctness
- Accuracy
- Exploration of different methods
- Simulation Results
- Presentation of Results
- Clarity
- Analysis and Evaluation

Reference

[1] https://en.wikipedia.org/wiki/Mastermind_(board_game)
[2] Mastermind Game Online - https://webgamesonline.com/mastermind/
[3] https://lirias.kuleuven.be/bitstream/123456789/184247/2/mastermind.pdf
[4] https://puzzling.stackexchange.com/questions/546/clever-ways-to-solve-mastermind
[5] Electronic Mastermind - https://americanhistory.si.edu/collections/search/object/nmah_1305800
[6] Electronic Mastermind - https://play.google.com/store/apps/details?id=com.elangames.electronicmastermind&hl=en&gl=US