***Smart Systems Final Project***

***Chess Using Genetic Algorithm***

* ***Group Members:***

1. Omar Mahmoud Nabil (Team Leader) [23012090]
2. Roshan Mohamed Helmy [23012096]
3. Malak Alaa ElDin Abdelhamid [23012095]
4. Hossam Khairy Mohamed Eid [23012099]
5. Omar Medhat Abdelhady [23012218]
6. Ziad Yakout Ibrahim [23012100]
7. Ahmed Mohamed Abdelgelil [23012102]

---------------------------------------------------------------------------***Chess Using Genetic Algorithm:*** chess using genetic algorithm combines the strategy of chess with the computational powers of Evolutionary Computation.

In chess, GA can be employed to develop algorithms capable of decision-making in the chess board. The algorithm selects, mutates, and recombines to achieve improved gameplay. It's designed with two options of your choice:  (1. Ai mode) or (2. Two players mode)

Building a chess game using Ai Genetic Algorithms requires huge and detailed steps of implementation, described as follows.

* ***Implementation of Board & Pieces:***

As the initial step, an implementation of the chess board is done with a default size of 8x8. The board’s colors are also implemented accordingly (white & beige).

Subsequently, we implement each piece of the chess (Pawn,Rook,Knight,Bishop,Queen,King) displaying their images in a clear visualized way and adding the valid and prescribed moves of every pieces. All possible moves are stored in an array and before a piece can move a condition occurs to check if the move is valid, not valid, or will face a checkmate. Additionally, you can press the button “R” on your keyboard to reset the game and start from the beginning.

* ***Implementation of Genetic Algorithm:***

Implementing GA helps evolve the game strategy. Starting off by initializing the algorithm parameters, number of strategies in each generation, and the initial population. Then, create random moves (up to 10 per game) and store them in a list. Fitness is evaluated based on the resulting board state. The crossover process goes as choosing the most fit parents from the best 50% with the best moves and combining them to create the child strategy. Subsequently, we apply the existing moves after the crossover to a new board and that's how the mutation process starts to take place. Mutation continues until finding the best output. Finally, we update the population with the best fitness and return the best move for the Ai to implement and play the best move independently according to the fitness and mutations calculations.

* ***Implementation of special moves:***

In chess, there are special moves or rules any player applies in a real chess game. Implementing these moves gives the game a different aspect of fun and professionalism.

* ***Promotion:*** when a pawn reaches the opposite end of the board, it’s promoted to another piece (queen).
* ***Castling:*** when the king moves 2 squares towards the rook, and the rook jumps over to the square next to the king. This can only happen when neither piece has moved before.
* ***En Passant:*** when a pawn moves two squares forward from its starting position, and an opposing pawn could have captured it had it moved only one square. The capture must be made immediately on the next move.
* ***Checkmate Check:*** when a player evaluates their move to see if it will result in a checkmate on their next turn or their opponent's turn and if there is no other option than checkmate then the game is over.
* ***Details about Visualization:***

Valid moves must be seen or known by each player. To apply so, we implemented that the valid moves of each piece that can move through it is visualized by a green square. Adding to that, the motion of pieces is animated so that the player can see from where the piece moved and the place it landed (moving through all squares to see the “from-to” path).

When a team wins a message appears in front of the player “BLACK/WHITE WIN”. An alternative message “DRAW” appears when both teams have the same winning situation.

* ***Libraries Used:***
* **Pygame** (library for creating 2D games)
* **Random** (library for random selection of moves and strategies)
* **Chess** (library to manage game states and moves)