# Chess Puzzle solver

**Github Link:** https://github.com/TylerMackJ/chess-puzzle-solver

**Goal:** The website chesspuzzle.net can generate chess puzzles for the user.  A chess puzzle is  a chess game in a state which can be won in X moves. By using the filter button on the website we are able to specify how many moves you can win the game in. Our project is based around this idea and implements the abilities to solve these puzzles in a timely manner by entering a puzzle in the form of a string that specifies the piece and location on the board, and brute forces white to win the puzzle. It then prints out the list of moves required to win that puzzle.

**Potential Users:**

Cheaters - A cheater of the game is able to run the program to find the correct moves to win the puzzle.

Chess puzzle builder - A chess puzzle builder is able to create puzzles and run the program on it to be sure there is a way to win it.

An algorithm developer -Due to the fashion in which we made the classes, a developer is able to use functionalities created to make game objects and move pieces around the board. This could allow someone to create a different algorithm to win the puzzle.

A puzzle filtering system - A website like the one used to test this program chesspuzzle.net may allow users to submit their own puzzles onto the website. This program could be used to simulate a win for testing purposes.

**Implemented Functionalities:**

getMoves - Takes in a board and returns a linked list with all the possible valid moves for each piece in a board. Each piece has specific ways it can move depending on the current configuration of the board. For example, the King can move one space in any direction while a knight can move in an L-shape with two spaces in one direction and one space in another.

populateBoard - Takes in a configuration of a board as a string and places the pieces in a 2-dimensional array, representing the chess board object. This can be used for many implementations such as making a chess game or in our case solving a chess puzzle.

bruteForce - Contains methods that take in the current puzzle and determines if a movement by white can cause a checkmate no matter what moves black had performed. This move is then printed for the user and is made on the board. This is then repeated until black is forced into a checkmate. This can be used to solve chess puzzles.

makeMove - This takes a game and a move and will return the game with the move made. This includes attacking other pieces. This can be used for implementing a chess playing system.

generatePuzzle - This creates an empty game and fills it with pieces randomly until it is possible to win the game in n moves. This can be used to create puzzles.