REPORT

Name: Marrium Jilani (20k-1748)

This report contains a brief explanation of the steps involved in the program and the results obtained.

- Firstly, I have create a chessboard using 2D matrix. The naming convention of the pieces is as such that black and white pieces have been labelled to begin with B and W respectively.
- Next, I have a function called **get_positions(move)** that will take the moves in form of "A6A7" format and convert it into row and column indices. Here, in "A6A7" A6 will be the position of the piece, and A7 will be the position it is moving to.
- The **is_in_check()** function is used as a helper function to check if a King can move to a certain position. If moving a piece to a certain position will cause it to be compromised, a checkmate situation will arise. This function will check all the pieces on the board to make sure none of them can move to the position the king is moving to. If any of them can, then function will return True.
- The **move_piece()** function is used to make a move on the board. However, before making the move it checks if the move is valid or not using **can_move()** function.
- The **can_move()** function will check whether the move is valid for the given piece or not. Following are a few checks that I have implemented:
    - Checking if the target position is on the board
    - Checking if the piece is moving to its own position.
    - Checking proper blocking
    - Checking valid moves for pawn movement including the situation when it is on its original position. Also ensuring that pawn kills only diagonally.
    - Checking valid moves for rook movement including the proper blocking.
    - Checking valid moves for knight movement including the proper blocking.

    - Checking valid moves for bishop movement including the proper blocking.
    - Checking valid moves for bishop movement including the proper blocking.
    - Checking valid moves for queen movement including the proper blocking.
    - Checking valid moves for king movement including the checkmate situation.

- If the move is valid, then **move_piece()** function will make the move.
- The computer's move is calculated by **minimax_alpha_beta()** which uses alpha beta pruning to give the best move. The depth of the solution has to be passed too. This function also requires some helper functions.
    - **Get_all_moves()** will get all possible moves for all the pieces for that color.
    - **Get_all_possible_moves()** will return all possible moves for that box.
    - **Evaluate_board()** will return the score for the moves based on some criteria. These include value of pieces on the board, mobility of pieces and control of center. The value of pieces is according to their importance. Based on this score, best move will be determined.
- The **get_user_move()** takes input from user. The user will need to enter the piece it wants to move and the move in the appropriate format.
- The **get_computer_move()** will generate the best move accordingly using alpha beta pruning.
- The **is_checkmate()** function will be a stopping criteria for the game. It checks that is the king in a checkmate position. If it is, game ends and a winner is declared.
- The **is_stalemate()** function will also be a stopping criteria for the game. It will check all moves for the player considering possibility for checkmate after making the moves. If the function returns true, the game_over flag will be set to true and game will be over.
- The computer is white and the human is black. Both players will take turns to make moves. The chessboard will be updated with their valid moves. The loop will keep on going unless the game is over.