Shayaan Hasnain

20I-0647

Section A

AI – Assignment 3

Explanation

- The create_chess_board function creates a 2D list representing the starting position of the pieces on a chess board. It first creates an empty 8x8 list of spaces.

It then places the white pieces in their starting positions on the first two rows of the chess board: rooks on a1 and h1, knights on b1 and g1, bishops on c1 and f1, queen on d1, king on e1, and pawns on each square of the second row.

It also places the black pieces in their starting positions on the last two rows of the chess board: rooks on a8 and h8, knights on b8 and g8, bishops on c8 and f8, queen on d8, king on e8, and pawns on each square of the seventh row.

The function then returns the 2D list representing the starting position of pieces on a chess board.

- The function move_validity_check is used to check the validity of a move made by a player on a given chess board. It takes in six parameters - the chess_board which is a 2D list representing the current state of the chess board, the starting_row and starting_col which represents the row and column indices of the piece that the player is trying to move, the ending_row and ending_col which represents the row and column indices of the position where the player is trying to move the piece to, and player which is a string representing the color of the player making the move.
  - The function checks the following conditions to determine if the move is valid:
  - The starting position is not empty.
  - The destination position is not occupied by a friendly piece.
  - The move is valid for the type of the piece being moved.

If all of these conditions are satisfied, the function returns True indicating that the move is valid, otherwise it returns False.

To check if the move is valid for the type of the piece being moved, the function calls the appropriate move validity function based on the type of the piece being moved - pawn, rook, knight, bishop, queen or king. Each of these move validity functions checks if the move is valid for that type of piece and returns True or False accordingly.

- The move_piece() function moves a piece on the chess board from a starting position to an ending position.

The input parameters are:

chess_board: a 2D list representing the current state of the chess board.

starting_row: an integer representing the row index of the starting position of the piece.

starting_col: an integer representing the column index of the starting position of the piece.

ending_row: an integer representing the row index of the ending position of the piece.

ending_col: an integer representing the column index of the ending position of the piece.

The function first retrieves the piece at the starting position of the chess board and stores it in a variable piece. It then sets the value of the starting position on the chess board to an empty space (' ') and sets the value of the ending position to the retrieved piece.

Finally, the function calls the display_board function to display the updated chess board.

- The check_checkmate function takes a chess board and the player's color as input and returns True if the player is in checkmate, and False otherwise. It first checks if the player is in check by calling the is_check function. If the player is not in check, it returns False. If the player is in check, the function checks if any of the player's pieces have any valid moves that would result in the player not being in check anymore. It does this by looping through all the pieces on the board and checking if each piece can be moved to any valid square that results in the player not being in check anymore. If it finds at least one such move, it returns False. Otherwise, it returns True, indicating that the player is in checkmate.

- The is_check function takes a chess board and the player's color as input and returns True if the player is in check, and False otherwise. It first finds the position of the player's king on the board. It then checks if any of the opponent's pieces can capture the king by looping through all the pieces on the board that belong to the opponent and checking if each piece can move to the square occupied by the player's king. If it finds at least one such piece, it returns True. Otherwise, it returns False.

- The check_checkmate function checks if a player is currently in checkmate, meaning that their king is in check and they have no valid moves to get out of check. It does this by first checking if the player is currently in check, and then simulating all possible moves for the player to see if any of them result in a state where the player is not in check. If there are no such moves, the player is in checkmate.

- The check_stalemate function checks if a player is in a stalemate, meaning that they are not currently in check but have no valid moves to make. It does this by simulating all possible moves for the player to see if any of them result in a state where the player is not in check. If there are no such moves, the player is in stalemate.
- The evaluate_board function evaluates the current state of the chess board and returns a score representing how advantageous the current position is for the given player. Each piece on the board is assigned a value based on its type (pawn, knight, bishop, rook, queen, or king), and the total score is calculated as the sum of the values of all of the player's pieces minus the sum of the values of all of the opponent's pieces.
- The get_all_moves function generates a list of all possible moves for a given player on the current chess board. It does this by iterating over all of the player's pieces and simulating all possible moves for each piece. The function returns a list of tuples, where each tuple represents a possible move as a pair of coordinate tuples (start, end).

- minimax_alpha_beta ()

The function takes in four arguments:

chess_board: a 2D list representing the current state of the chess board.

player: a string representing the player whose turn it is to make a move.

depth: an integer representing the maximum depth of the search tree.

alpha and beta: initial values for alpha and beta, which are used for pruning.

The function first checks if the maximum depth has been reached or if a checkmate or stalemate has been reached. If so, it returns the evaluation score of the current board. If not, it generates all possible moves for the current player and recursively evaluates each move by calling minimax_alpha_beta with the new state of the chess board and the opposite player.

If maximizing_player is True, the function will try to maximize the score by selecting the move with the maximum evaluation score (i.e., the best move for the current player). The function keeps track of the maximum score found so far in the max_eval variable and updates the alpha value if the new score is greater than the current alpha. If beta becomes less than or equal to alpha, the function breaks out of the loop since it is no longer necessary to evaluate the remaining moves. The function returns the maximum score found.

If maximizing_player is False, the function will try to minimize the score by selecting the move with the minimum evaluation score (i.e., the best move for the opposite player). The function keeps track of the minimum score found so far in the min_eval variable and updates the beta value if the new score is less than the current beta. If beta becomes less than or equal to alpha,

the function breaks out of the loop since it is no longer necessary to evaluate the remaining moves. The function returns the minimum score found.

- AI_moves() uses the Minimax algorithm with Alpha-Beta pruning to find the best move for the AI player given the current state of the chess board, while user_moves() asks the user to input their move and checks whether the move is valid.

AI_moves() first sets the maximum depth for the search and initializes the best move and its score to None and negative infinity, respectively. It then generates all possible moves for the current player using the get_all_moves() function and loops over each move. For each move, it makes a copy of the chess board and simulates the move by calling move_piece(). It then evaluates the move using the minimax_alpha_beta() function with the given parameters, which returns a score for the move. If the score is better than the best score so far, the function updates the best move and its score. Finally, the function displays the current state of the chess board and returns the best move.

- user_moves() uses a while loop to repeatedly ask the user to enter their move until a valid move is entered. It then parses the move to obtain the starting and ending positions in terms of row and column indices, checks whether the move is valid using the move_validity_check() function, displays the current state of the chess board, and returns the starting and ending positions.