Artificial Intelligence Game

This code is for a two-player board game that is played on a 7x7 grid. Each player has a piece on the board and can make two moves per turn. The first move is to move their own piece to an empty square on the board. The second move is either to move their own piece to another empty square or to remove an opponent's piece from the board. The game ends when one player reaches the other end of the board with their piece, or when all of the squares on the board have been removed.

The code consists of several functions that handle different aspects of the game. The `initBoard` function initializes the board with empty squares and places the players' pieces on the board. The `printBoard` function prints the current state of the board to the console. The `makeMove` function is used to move a player's piece to a new square on the board. The `removeSquare` function is used to remove a square from the board. The `writeMoveToFile` function is used to write the details of a player's move to a file. The `readMoveFromFile` function is used to read the details of a player's move from a file.

The main game loop is where the game is played. The loop alternates between the two players, with the AI player going first. On each turn, the AI player uses the `minimax` function to determine the best move to make. The `minimax` function uses the Min-Max algorithm to explore all possible moves and choose the one that maximizes the AI player's score while minimizing the opponent's score.

The player's turn consists of two moves. The player can either move their piece to a new square on the board or remove an opponent's piece from the board. The player enters the coordinates of their first move and the code updates the board and player's position accordingly. The player then enters the coordinates of their second move, which can either be another move for their own piece or a request to remove a square from the board. The code updates the board and player's position as needed.

The game ends when one player reaches the other end of the board with their piece, or when all of the squares on the board have been removed. When the game ends, the code prints the winner to the console and the game loop exits.

Details on the minimax function

The `minimax` function is a recursive function that is used to determine the best move for the AI player to make. It uses the Min-Max algorithm, which is a decision-making strategy used in artificial intelligence and game theory.

The Min-Max algorithm is a way to determine the best move for a player in a two-player, zero-sum game. In a zero-sum game, the sum of all payoffs is zero, meaning that one player's gain is equal to the other player's loss. In other words, one player's gain is exactly offset by the other player's loss.

The idea behind the Min-Max algorithm is to explore all possible moves and their consequences, and choose the move that maximizes the player's score while minimizing the opponent's score. The algorithm does this by recursively evaluating the scores of all possible moves, starting from the current state of the game.

The `minimax` function takes as input the current state of the board, the player's ID (either 1 or 2), and two values called `alpha` and `beta`. These values are used in the algorithm to keep track of the best scores that have been found so far.

The function first checks if the game is over, and if it is, it returns the score for the current player. If the game is not over, the function iterates over all possible moves and calls itself recursively for each move, passing in the updated board state and the opposite player's ID.

The function then determines the best score for the current player based on the scores of the possible moves. If the current player is player 1 (the AI player), it will choose the move with the highest score. If the current player is player 2, it will choose the move with the lowest score.

The function returns the best score for the current player, which can then be used by the main game loop to decide which move to make.