Report:

For clarification player and opponent are referring to player 1 and player 2 I explained it like that to match up more closely with variables in my program.

Intro:

My engine uses the minmax without alpha beta pruning search in order to determine the value of any move the engine is going to make. I've set my engine to do a depth search of four as any more than that causes slow downs on my laptop and it accomplishes what is required.

Evaluation Function:

How the board is parsed:

The gameboard is parsed by checking each row, column and diagonal on the board to check for moves and how close they are to a connect 4 aka a win.

Evaluating Moves:

The evaluation function works by categorising "moves". A move is simply a object that is a potential connect 4 win. For example it would store owner count which is the amount of symbols the owner has in a row that can make a connect 4, meaning that an opponents piece can't be blocking the creation of a connect 4. Based on the value of owner count determines the score it's worth. For example 1 would only equal a score of 1 while having a owner count of 2 would be worth 5 and an owner count of 3 would be worth 10.

These values are set the way they due to trial an error as obviously the closer you are to a connect 4 the more valuable that move would become and by extension the board state as a whole.

After each of these values are generated they are stored into an object called MovesContainer. Both the opponent and player have one which stores their moves on the board.

The MovesContainer is also responsible for determining when new moves should be created. It does this by reading in all the values from the board and uses that to make the decision. It'll stop making a move if it reads in a value of the other player and remove the Move that was being created as it has no value being calculated in the final evaluation.

The amount of freespace is also recorded prior to the players piece and then recorded separately afterwards This essentially allows the freespace after a row of player pieces to be counted onto the freespace of the next move. This was needed to be done as only one move is counted at a time.

After the board state has been finished analysing, the MovesContainer returns the total score of all the moves. The total for the player is then subtracted by the opponents value. Meaning that if they are even the board state would be 0, if the opponent is winning then the score would be more negative, and if the player was winning then the score would be more positive. The value generated is then returned and used in the minimax algorithm to pick the most appropriate move.

Evaluating a draw, win or loss:
To determine if a either of these board states have occured my evaluation function just checks if either the opponent or player have had a 4 in a row anywhere. This is done by having a counter called playerwin and opponent win that increment by one when there piece is being parsed but are reset to 0 if it is a blank space or the other players piece.
Once a value of 4 has been found it'll return 1000 if it's the players winor -1000 if it's the opponents win.

To determine if it's a draw at the end of the evaluation function all the spots are checked to see if there are any empty spots, if there are not then a score of 0 is returned as both players performed as well as each other.

Perft Function:

My perft function works by counting all the nodes as the tree is being created, then returning that value. This doesn't have any connection to the minmax function.