**TEST DOCUMENT**  
The objective with this project was to implement a two-person game where:  
1. The user plays against computer.  
2. The computer has a strategy and uses a depth first search (the minimax algorithm) to find a good move.  
  
The game implemented is a generalized version of tic-tac-toe. I.e., the user determines the size of the board but otherwise the rules are the same as for a normal 3x3 tic-tac-toe.  
  
**ABOUT THE MINIMAX ALGORITHM:**

The minimax algorithm is essentially a decision rule for a player X about to make a move on the board. The underlying idea is to *minimize the possible loss for the scenario where the opponent is making optimal moves*. It is therefore a "pessimistic and safe" way of playing,  
which is the exact reason for its effectiveness.  
  
A player X employs the minimax algorithm with the following steps:   
- Look ahead a number of moves given the current game state, thus generating a game tree with all possible future game states.   
- A state where the game has reached the end is becoming a leaf and is assigned + if its a winning   
outcome, - if it is a losing outcome, 0 if draw.   
- The value of a parent node is assigned through backtracking, i.e. it is determined based on the children’s values.  
 1. Either the **maximum** value of the children are picked for the parent (if children states are outcomes of player X's moves).   
 2. Or the **minimum** (if children states are outcomes of opponent’s move).  
- Pick the child of the root (initial game state) with the highest value.  
  
A player who employs the minimax algorithm as described above is indeed a "perfect player" meaning he is  
guaranteed to at least tie. Why does the minimax work? The crucial part is that the player always picks the  
move that *minimizes his possible loss*. That is to say, we assume our opponent will always pick the move that results in the worst score for us. However, we don't know for sure this will indeed be the case, but we assume so; we are "minimizing the possible loss".  
If the opponent doesn't end up playing to our greatest disadvantage, well then great. This just means we can get to a favorable position sooner.  
  
However, the above algorithm needs some modifications to be usable for games that are not as simple as tic-tac-toe. When the game tree becomes big enough it is no longer feasible to get all the way down to the end states of the game. In this case we need to:

1. Determine a depth, i.e., a limit for how far down in the game tree we can go.

2. Have an evaluation function that calculates the *chances of winning* for a given state. Using the function, a heuristic value is assigned to the leaf game states.

It now becomes more interesting because we can start talking about the player having an actual strategy (meaning the evaluation function). If the player’s evaluation function is really good and he can go fairly deep, then he can get pretty close to being “perfect”. Coming up with a function that calculates the chances of winning accurately however is, simply put, hard. The more complex of a game, the more factors to take into account. In chess, for example, only a few factor would be general structure of the game state, the number and kind of black and white pieces, pawn formation, mobility (Shannon 5).

**ABOUT THE IMPLEMENTED PROGRAM:**

Luckily tic-tac-toe is not nearly as complex as chess and therefore a relatively effective evaluation function can be implemented fairly easily. The evaluation function used in the program looks at how many rows /columns /diagonals that don’t contain more than one type of token. If we are looking from player X’s perspective, for each such row /column /diagonal with only X in them, we add up the number of X’s and for those with only O’s in them we subtract the number of O’s. (A game state that is an end state is automatically assigned the heuristic + if winning state, - if losing state, and 0 if draw).

To give an example, the following game state would have a heuristic value of 0.

-------------

| X | X | - |

-------------

| O | O | X |

-------------

| - | - | O |

-------------

**TESTS:**

The tests consisted in playing tic-tac-toe games of various sizes and looking at how cleverly the computer played against us. The depth used for all cases was 3.

**Case 1:** Playing to win against the computer (us playing perfectly).

**Result**: The computer plays perfectly.

Human's turn: enter (row, column)

1 1

-------------

| X | - | - |

-------------

| - | - | - |

-------------

| - | - | - |

-------------

Intelligent Computer's turn:

-------------

| X | - | - |

-------------

| - | O | - |

-------------

| - | - | - |

-------------

Human's turn: enter (row, column)

1 2

-------------

| X | X | - |

-------------

| - | O | - |

-------------

| - | - | - |

-------------

Intelligent Computer's turn:

-------------

| X | X | O |

-------------

| - | O | - |

-------------

| - | - | - |

-------------

Human's turn: enter (row, column)

3 1

-------------

| X | X | O |

-------------

| - | O | - |

-------------

| X | - | - |

-------------

Intelligent Computer's turn:

-------------

| X | X | O |

-------------

| O | O | - |

-------------

| X | - | - |

-------------

Human's turn: enter (row, column)

2 3

-------------

| X | X | O |

-------------

| O | O | X |

-------------

| X | - | - |

-------------

Intelligent Computer's turn:

-------------

| X | X | O |

-------------

| O | O | X |

-------------

| X | O | - |

-------------

Human's turn: enter (row, column)

3 3

-------------

| X | X | O |

-------------

| O | O | X |

-------------

| X | O | X |

-------------

IT'S A TIE!

**Case 2:** Playing slightly differently, again we play perfectly and try to win.

**Result**: The computer plays perfectly.

Human's turn: enter (row, column)

1 1

-------------

| X | - | - |

-------------

| - | - | - |

-------------

| - | - | - |

-------------

Intelligent Computer's turn:

-------------

| X | - | - |

-------------

| - | O | - |

-------------

| - | - | - |

-------------

Human's turn: enter (row, column)

1 3

-------------

| X | - | X |

-------------

| - | O | - |

-------------

| - | - | - |

-------------

Intelligent Computer's turn:

-------------

| X | O | X |

-------------

| - | O | - |

-------------

| - | - | - |

-------------

Human's turn: enter (row, column)

3 2

-------------

| X | O | X |

-------------

| - | O | - |

-------------

| - | X | - |

-------------

Intelligent Computer's turn:

-------------

| X | O | X |

-------------

| O | O | - |

-------------

| - | X | - |

-------------

Human's turn: enter (row, column)

2 3

-------------

| X | O | X |

-------------

| O | O | X |

-------------

| - | X | - |

-------------

Intelligent Computer's turn:

-------------

| X | O | X |

-------------

| O | O | X |

-------------

| - | X | O |

-------------

Human's turn: enter (row, column)

3 1

-------------

| X | O | X |

-------------

| O | O | X |

-------------

| X | X | O |

-------------

IT'S A TIE!

**Case 3:** We make a mistake early in the game (by giving the computer the chance to create a fork).

**Result**: The computer capitalizes on the mistake and wins. Computer plays perfectly.

Human's turn: enter (row, column)

2 3

-------------

| - | - | - |

-------------

| - | - | X |

-------------

| - | - | - |

-------------

Intelligent Computer's turn:

-------------

| - | - | O |

-------------

| - | - | X |

-------------

| - | - | - |

-------------

Human's turn: enter (row, column)

2 1

-------------

| - | - | O |

-------------

| X | - | X |

-------------

| - | - | - |

-------------

Intelligent Computer's turn:

-------------

| - | - | O |

-------------

| X | O | X |

-------------

| - | - | - |

-------------

Human's turn: enter (row, column)

3 1

-------------

| - | - | O |

-------------

| X | O | X |

-------------

| X | - | - |

-------------

Intelligent Computer's turn:

-------------

| O | - | O |

-------------

| X | O | X |

-------------

| X | - | - |

-------------

Human's turn: enter (row, column)

1 2

-------------

| O | X | O |

-------------

| X | O | X |

-------------

| X | - | - |

-------------

Intelligent Computer's turn:

-------------

| O | X | O |

-------------

| X | O | X |

-------------

| X | - | O |

-------------

Intelligent Computer WON!

**Case 4:** A 5x5 tic-tac-toe is less interesting because it’s pretty hard to lose or win. That is to say, it’s hard for either player to be “clever”. However, the computer does what is expected; it blocks our attempt to win when necessary and capitalizes on our mistakes.

Human's turn: enter (row, column)

1 1

---------------------

| X | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

Intelligent Computer's turn:

---------------------

| X | - | - | - | - |

---------------------

| - | O | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

Human's turn: enter (row, column)

1 5

---------------------

| X | - | - | - | X |

---------------------

| - | O | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

Intelligent Computer's turn:

---------------------

| X | - | - | - | X |

---------------------

| - | O | - | - | - |

---------------------

| - | - | O | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

Human's turn: enter (row, column)

1 2

---------------------

| X | X | - | - | X |

---------------------

| - | O | - | - | - |

---------------------

| - | - | O | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

Intelligent Computer's turn:

---------------------

| X | X | O | - | X |

---------------------

| - | O | - | - | - |

---------------------

| - | - | O | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | - |

---------------------

Human's turn: enter (row, column)

5 5

---------------------

| X | X | O | - | X |

---------------------

| - | O | - | - | - |

---------------------

| - | - | O | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | X |

---------------------

Intelligent Computer's turn:

---------------------

| X | X | O | - | X |

---------------------

| O | O | - | - | - |

---------------------

| - | - | O | - | - |

---------------------

| - | - | - | - | - |

---------------------

| - | - | - | - | X |

---------------------

Human's turn: enter (row, column)

4 5

---------------------

| X | X | O | - | X |

---------------------

| O | O | - | - | - |

---------------------

| - | - | O | - | - |

---------------------

| - | - | - | - | X |

---------------------

| - | - | - | - | X |

---------------------

Intelligent Computer's turn:

---------------------

| X | X | O | - | X |

---------------------

| O | O | O | - | - |

---------------------

| - | - | O | - | - |

---------------------

| - | - | - | - | X |

---------------------

| - | - | - | - | X |

---------------------

Human's turn: enter (row, column)

3 5

---------------------

| X | X | O | - | X |

---------------------

| O | O | O | - | - |

---------------------

| - | - | O | - | X |

---------------------

| - | - | - | - | X |

---------------------

| - | - | - | - | X |

---------------------

Intelligent Computer's turn:

---------------------

| X | X | O | - | X |

---------------------

| O | O | O | - | O |

---------------------

| - | - | O | - | X |

---------------------

| - | - | - | - | X |

---------------------

| - | - | - | - | X |

---------------------

Human's turn: enter (row, column)

5 4

---------------------

| X | X | O | - | X |

---------------------

| O | O | O | - | O |

---------------------

| - | - | O | - | X |

---------------------

| - | - | - | - | X |

---------------------

| - | - | - | X | X |

---------------------

Intelligent Computer's turn:

---------------------

| X | X | O | - | X |

---------------------

| O | O | O | O | O |

---------------------

| - | - | O | - | X |

---------------------

| - | - | - | - | X |

---------------------

| - | - | - | X | X |

---------------------

Intelligent Computer WON!From the tests we can conclude that the computer plays as cleverly as can be expected. Whereas it is somewhat hard to judge for boards of size bigger than 3x3, we can say that for the normal tic-tac-toe we have created a seemingly perfect player. This is however not surprising given that a game of tic-tac-toe is deterministic 3 levels down (which was the depth we used in the minimax algorithm) after only a few moves. It would be more interesting to implement a more complex game and even perhaps let computer players play against each other using different evaluation functions.