a) In coming up with this evaluation function, we choose to give higher values to those states where we win and then in decreasing order, those that have more open 3's, open 2's, and total number of ways to win for us. We give the same values to each of the opponents positions, just negative values. We also include a scaling factor for keeping our pieces towards the middle instead of the sides as the middle has an advantage to winning. We also put a lot of value towards setting traps and avoiding any enemy traps. (i.e. When there are two consecutive vertical positions where they/we can win)

b) value += 1000*usWin - 1000*themWin - 500*themCanWin - 750*avoid + 750*play + (25*numThrees + 50*usDoubleThrees - 25*themThrees - 50*themDoubleThrees + 10*numTwos - 10*themTwos + waysToWin) * widthFact;

- usWin = state where we won (1000 points)
- themWin = state where they won (-1000 points)
- themCanWin = number of enemy 3's with a winning position in the next move (-500 points)
- Avoid = number of positions where enemy has 2 different winning positions in 2 consecutive vertical spots (-750 points)
- Play = number of positions where we have 2 different winning positions in 2 consecutive vertical spots (750 points)
- numThrees = number of our 3's we have in a row (25 points)
- usDoubleThrees = number of our 3's with open positions on both sides (50 points)
- themThrees = number of 3's enemy has in a row (-25 points)
- themDoubleThrees - number of enemy 3's with open positions on both sides (-50 points)
- numTwos = number of 2's we have in a row (10 points)
- themTwos = number of 2's enemy has in a row (-10 points)
- waysToWin = total number of ways to win from every position that can be played (1 points)
- It is multiplied by widthFact which scales depending on where it is on the board (makes moves in the middle worth more)

c) - Total Score = 47 (Total Score is an int value)
    - waysToWin = 17 (1 point each)
    - numTwos = 3 (10 points each) * 1.5 (widthFact)
    - themTwos = 3 (-10 points each) * 1.75
    - numThrees = 1 (25 points) * 1.5

```
.......
.......
....X..
....X..
..XOO..
.XOXOO.
```

## Games w/ RandomAI - Won 5/5

```
.......   .......   .......   ....o..   .......
.......   .......   .......   ...xx..   .......
...x...   ...ox..   .......   x.oxo..   ...x...
...x...   ..xxx..   .o.....   o.xxx..   ..ox...
..ox...   .ooxx..   xxxx..o   oxxoo..   ..oxx..
..oxo..   .ooxxoo   ooxx..o   oooxx..   o.xxo.o
```

## Games w/ StupidAI - Won 5/5

```
.......   .......   .......   .......   .......
o......   o......   o......   o......   o......
x......   x......   x......   x......   x......
o......   o......   o......   o......   o......
o......   o......   o......   o......   o......
o.xxxx.   o.xxxx.   o.xxxx.   o.xxxx.   o.xxxx.
```

## Games w/ MonteCarlo- Won 10/10

```
XX.XX.X   XX.XX.X   XX.OOX.   XX.XX.X   .......
OO.OO.O   OO.OO.O   OX.OOX.   OO.OO.O   ...O...
XX.OO.X   XX.OO.X   XX.XXO.   OO.OO.X   ..OX.O.
XO.OOXX   XO.OO.X   OO.XXX.   XO.OOXX   ..XXXX.
OX.XXOO   OXXXX.O   XOXOOX.   XX.XXOO   ..XOOO.
OX.XOOX   OXOXOOX   OOOXXOO   OX.XOXX   X.OXXO.

XX.XX.X   .......   XX.XX.X   XX.OXO.   XX.XX.X
OO.OO.O   ...O...   OO.OO.O   OX.OOX.   OO.OO.O
OO.OO.X   ..OX.O.   OO.OO.X   XX.XXO.   XX.OO.X
XO.OOXX   ..XXXX.   XO.OO.X   OO.XXX.   XO.OOXX
XX.XXOO   ..XOOO.   XXXXX.O   XOXOOX.   OX.XXOO
OX.XOXX   X.OXXO.   OXOXOXX   OOOXXOO   OX.XOOX
```

## Total Games Won: 20/20

4. We implemented the alpha beta pruning without a successor function first, to see how much pruning would be done, and how much more depth we could add, without changing the order we added nodes. Then to increase the pruning, we changed our ordering of moves to test start in the middle of the board going towards the outside for MAX. We did this because typically the best moves as determined by our evaluation function are in the middle of the board, so we wanted to prune out unnecessary move possibilities on the edges. For the MINI function, we start from the outside and work our way to the middle, for the opposite reason. The final depth of the tree we can reach even with pruning is not the deepest we had tried, because we made our evaluation function slightly more complex. This complexity makes the score associated with each node a more accurate representation of the board state and thus a more correct measure of which branch to go down.