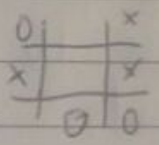
Treant.JS - creates tree diagrams

Mini Max

1. return a value if a terminal state is found (+10, 0, -10)
2. go through available spots on board
3. call the minimax function on each available spot (recursion)
4. evaluate returning values from function calls
5. and return the best value

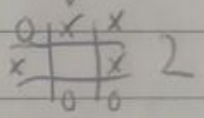
Code explanation

level 0
Turn x (AI)
Action: Maximise

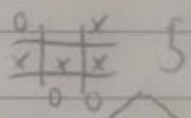


~~Score~~ Score = 10

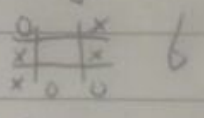
level 1
Turn O (human)
minimize



Score = -10

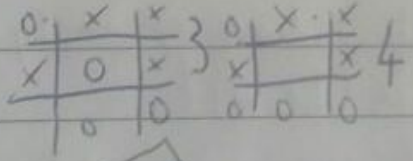


Score = 10

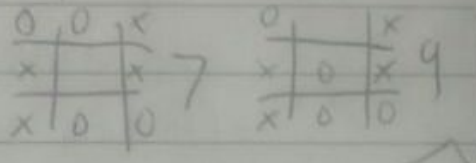


Score = -10

level 2
Turn X (AI)
Action minimize

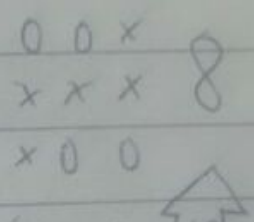


Score = -10 Score = -10



Score = -10 Score = -10

level 3



Score = 10

function ~~fill~~ winning(board, player)

James / Tahmoor

need to define the minimax algo function with two arguments newboard and player.
Find the indexes of the available spots in the board set it to variable called availSpots.

Code

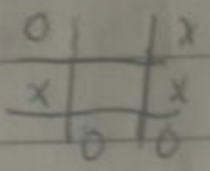
```
function minimax ~~(newboard~~ newboard, player){
// available spots
var availSpots = emptyIndexies(newBoard);
```

need to check for terminal states and return a value. If O wins should return -10, if X wins should return +10. If the length of the availableSpots array is zero, there is no more room resulting in a tie returning zero.

Code

```
if (winning(newBoard, humanPlayer){
    return {score: -10};
}
    else if (winning(newBoard, aiPlayer)){
    return {score: 10};
    }
else if (availSpots.length === 0){
        return {score: 0};
    }
```

for tic tac toe - define ~~the board~~ as an array with 9 items. Each item will have its index as a value.



using this board as an example

var board = ["O", 1, "X", "X", 4, "X", 6, "O", "O"];

Declare ai player and human variables set them to "X" and "O"

needs function that looks for winning combinations and returns true if found (James / Tahmoor) and a function that lists the indexes of available spots in the board

Code so far

```
var humanP = "O";
var aiplayer = "X";
// returns list of the indexes of empty spots on board
function emptyIndexies(board){
return board.filter(s => s != "O" && s != "X");
}
```

Need to collect the scores from each empty spots to evaluate later.
So make an array called moves and loop through empty spots while collecting each move's index and score in an object called move.