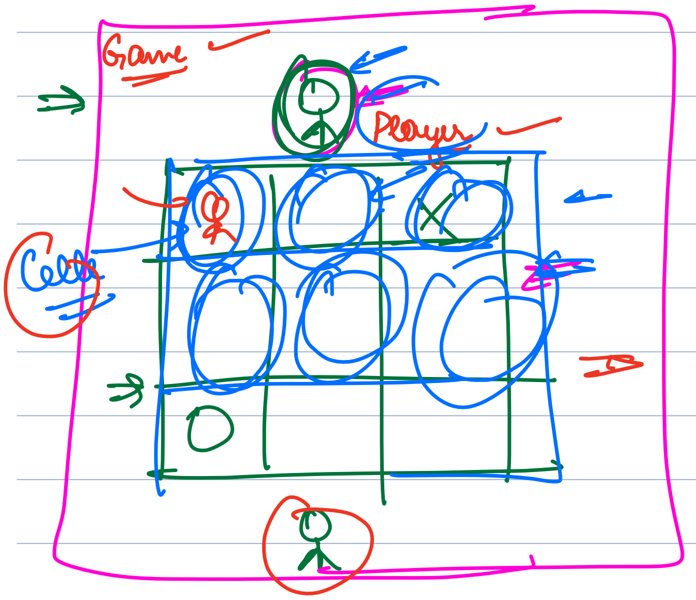
TicTac Toe 2

Will start
at 9:10PM

1. Class diagram
2. Finding winner in O(1)
3. Code (Start)

End TTT code in next class

CLASS DIAGRAM
- Nouns
- Visualization
- Entities
- Design Patterns

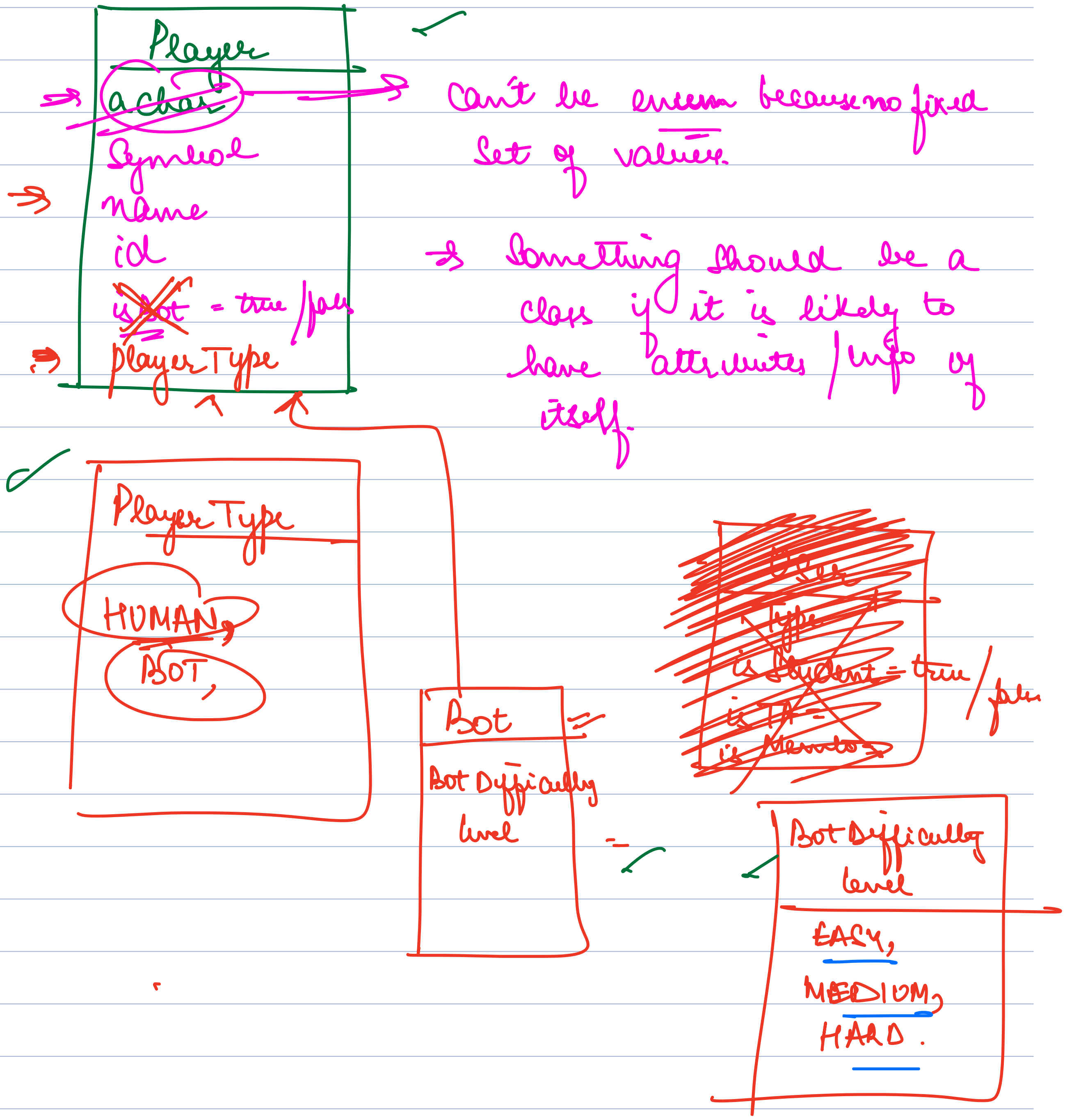


Game

List<Player>
Board.
List<Move>
Player winner;
Game State
next Player Move Idx

Game State

IN-PROG,
DRAW,
SUCCESS,
PAUSED,



Can't be enum because no fixed set of values.

→ Something should be a class if it is likely to have attributes / info of itself.

Board

- Size
- List<List<Cells>>

✓

Cell

- int row
- int col
- ~~isEmpty~~
- cell State
- Player

✓

→ null if cell is empty

→ else it will have player on the cell.

Cell State

EMPTY,

FILLED,

→ BLOCKED

filled

empty

blocked

FB biggest fear

→ google is creating a brilliant social network

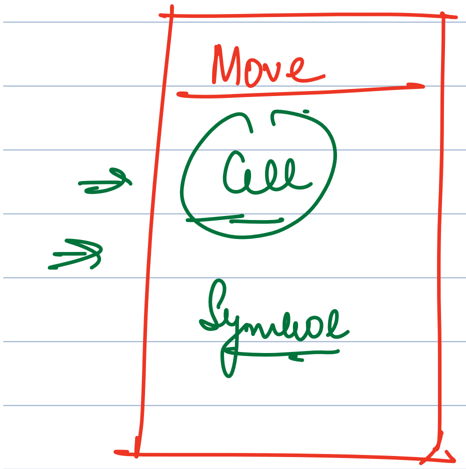
~~Google+~~

→ F/W like Spring Boot

→ F/W like React

→ kubernetes

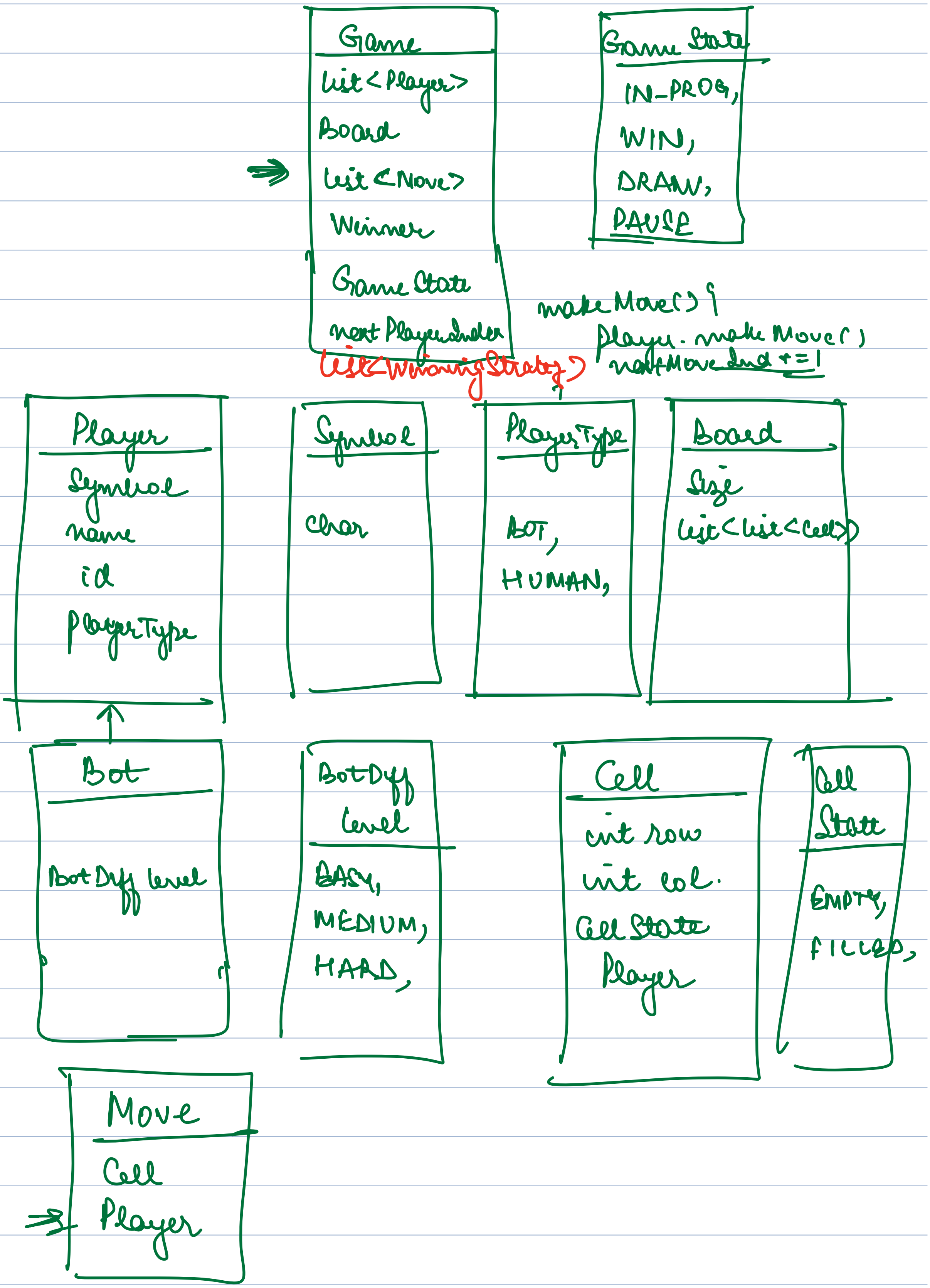
exit today

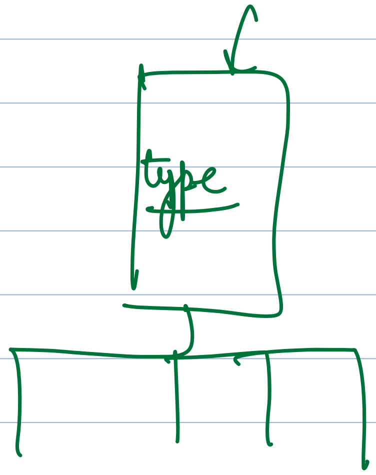


List <Board>

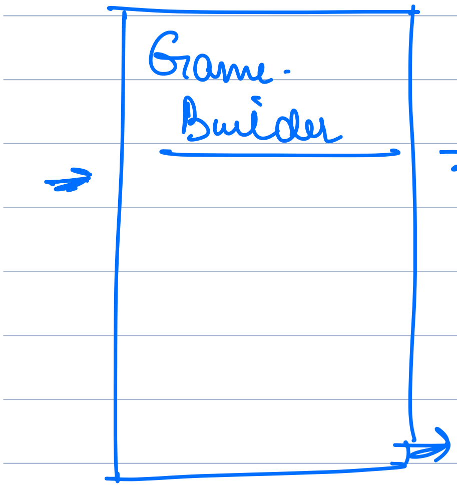
move → putting a symbol at a particular cell

CLASS DIAGRAM OF TIC TAC TOE





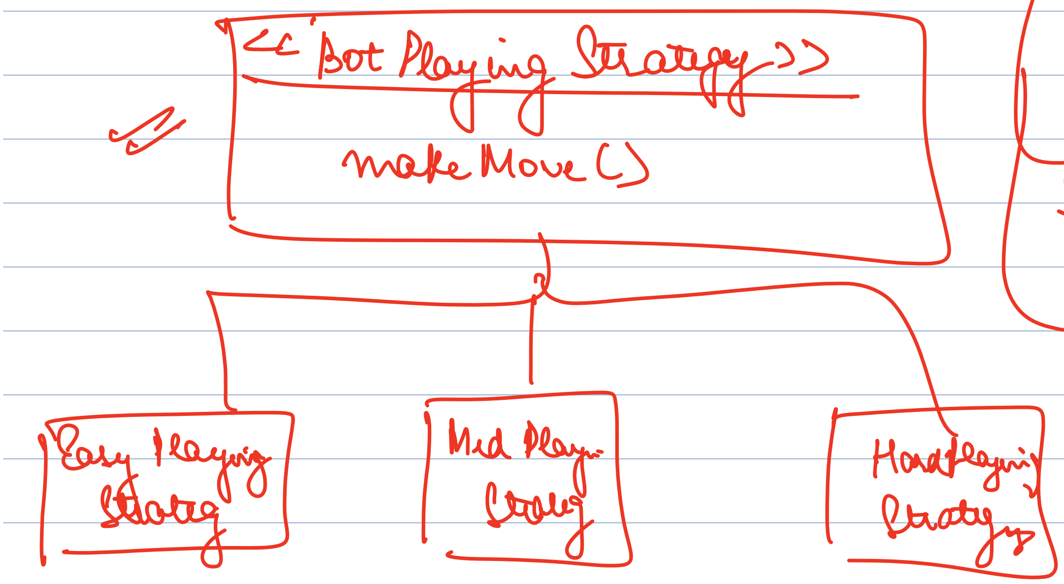
```
if (parent.type == ____)
    ____
    ____
    ____
elif (                  )
    ____
    ____
    ____
```



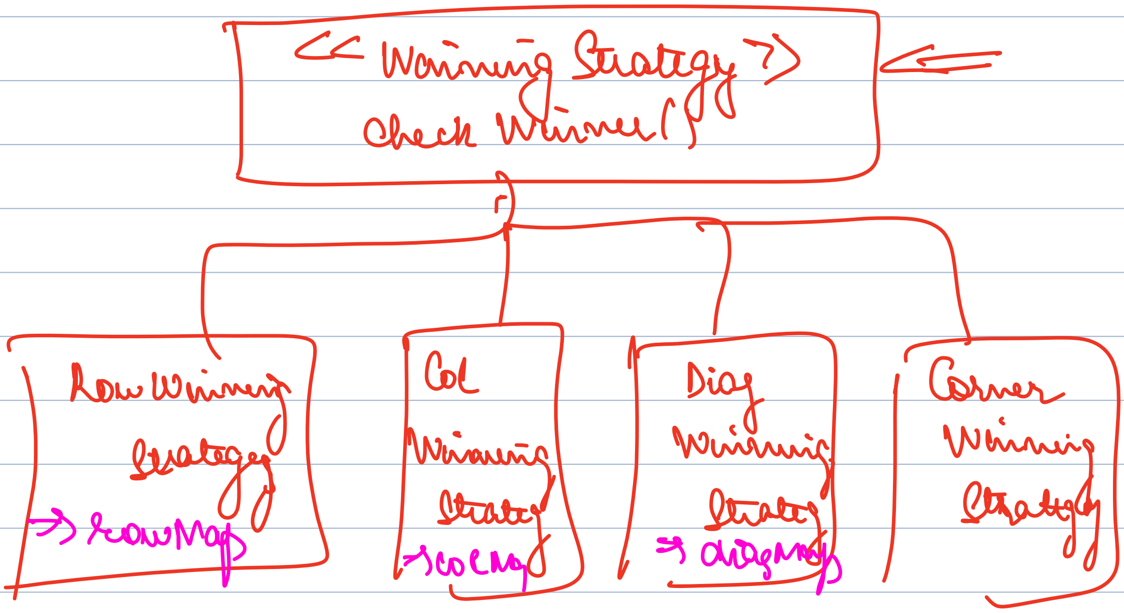
→ to validate if one symbol is there with only one player.

→ to validate Only 1 Bot

Dont create a singleton unnecessarily



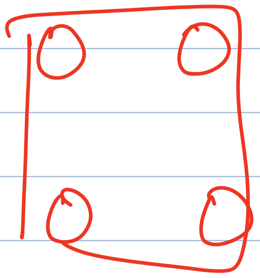
- Adapter ✓
- Factory ✓
- Strategy ✓
- Singleton ✓
- Builder ✓

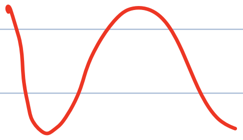


Bot Playing Strategy Factory

getBotPlayingStrategy forDiffLevel(____) {

}

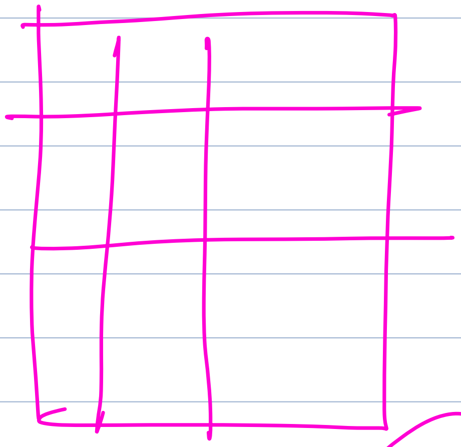




How to check if someone has won in O(1)

Assume victory happens:

- → same char over complete row
- → ______ col
- → ______ diagonals

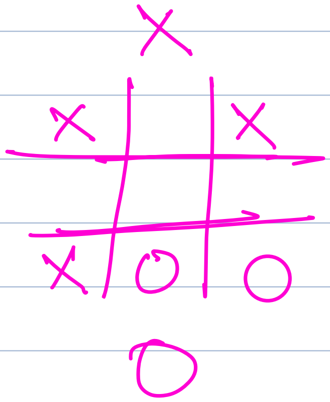


~~for every player (X) O(N)~~ w:

```
for int i = 0; i < n; ++i
    is sol = true
    for j=0; j<n; ++j
        if board[i][j] != p
            is_sol = false
            break
    if is_sol = true
        return true
```

$O(N^3)$

→ Naman made a move
Can praveen win?



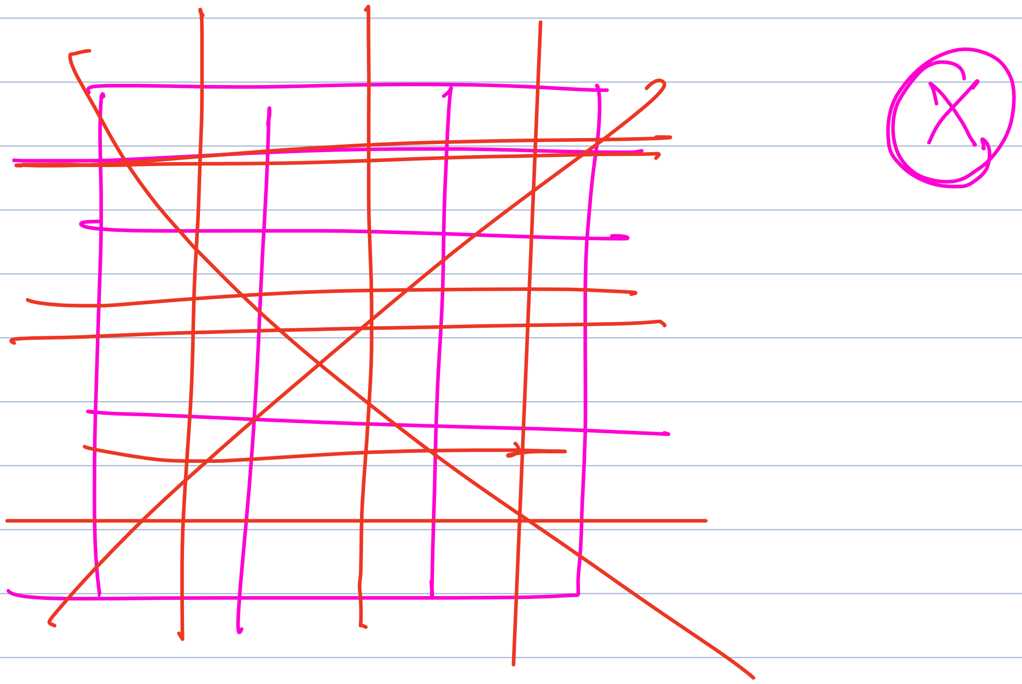
→ Only the player who last made the move can be winner.

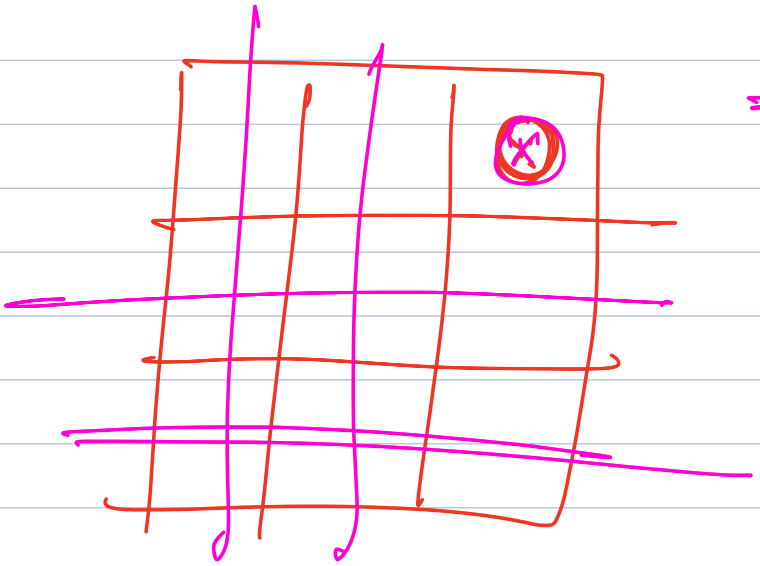
I.

```
Check Winner (Board) {

        → $\underline{\underline{O(N^3)}}$

}
```

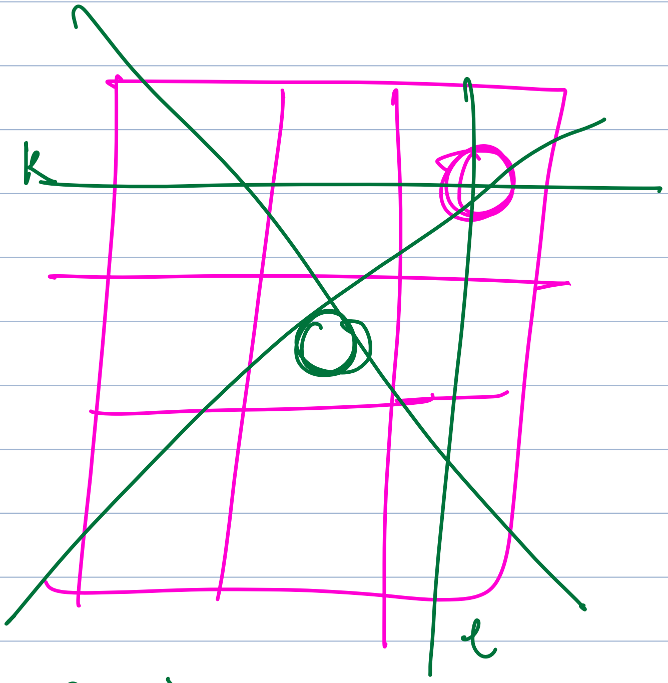
II.

```
checkWinner (Board, lastMovePlayer) {

        → $O(\underline{\underline{N^2}})$

}
```





→ after a particular move victory can only happen in row, colⁿ, diag containing the cell of that move



1 row
1 col
2 diag

O(N)

III Check Winner(Board, Player, cell (k, l))

```
O(N) → row  [ for (j=0; j<n; ++j)
                  if board[k][j] != player.symbol
                      = false
O(N) → col  [ for i=0; i<n; ++i
                  if board[i][l] != player.symbol
                      → false
```

O(N)

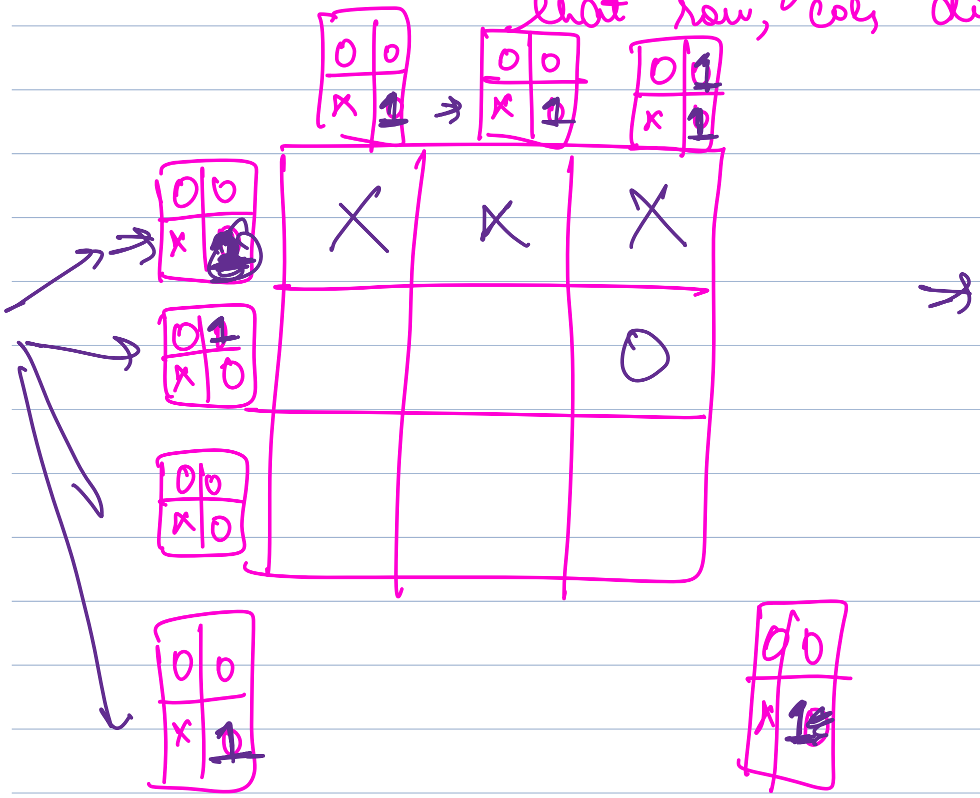
O(N)

++i faster than i++

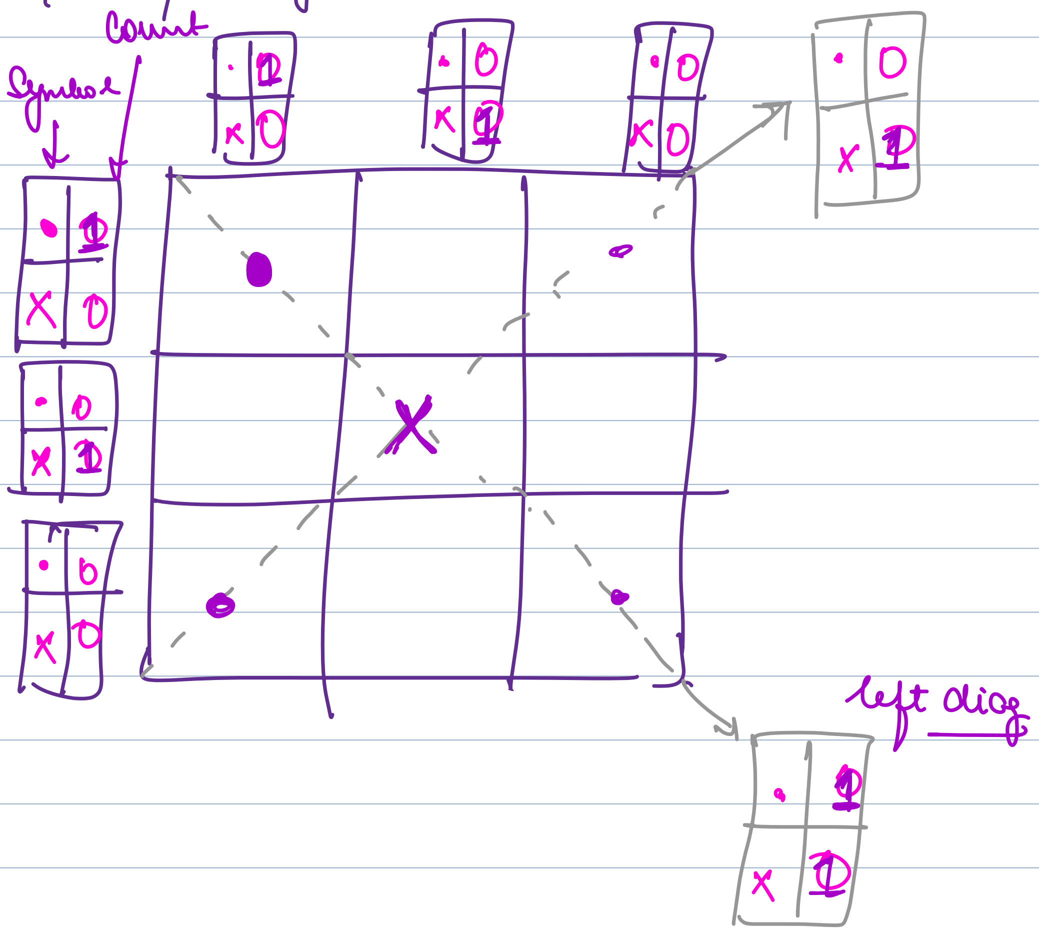
i++

IV → create a map for each row, each col, each diag

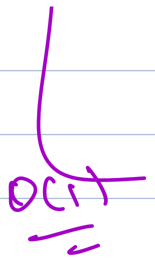
→ stores count of each symbol in that row, col, diag.



⇒ Map < Symbol, Integer > which stores count of occurrence of that symbol in that row / col / diag.



```
Check Winner( (i,j), X ) {
    ① update Map of Row i
    ② update Map of col j
    ③ if (i,j) is on left diag
            update Map of left
    ④ if (i,j) is on right diag
            update map of right diag            O(1)

    if Map of row i says X has a count of n ⇒ return true
       ___ col j ___ X ___ n ⇒ return tr
}
```



OCI