## GameBoard(int r, int c, int to\_win)

Input: State: Gameboard not	Output: GameBoard Object State: GameBoard Initialized	Reason: This test case is unique
Initialized		because it tests a value that is not a minimum or maximum size. It also checks the
r: 5 c: 5 to win: 3		GameBoard object that is used for every other test case besides the ones that test the
5		constructor
		Function: Constructor_NORM

## GameBoard(int r, int c, int to\_win)

Input: State: Gameboard not	Output: GameBoard Object State: GameBoard Initialized	Reason: This test case is unique
Initialized		because it tests the minimum values the constructor can
2		make a table with.
r: 3 c: 3		Function: Constructor_MIN
to_win: 3		

#### GameBoard(int r, int c, int to\_win)

Carrier and the control of the control							
Input:	Output:	Reason:					
State: Gameboard not	GameBoard Object	This test case is unique					
Initialized	State: GameBoard Initialized	because it tests the maximum values the constructor can					
	A GameBoard of 100x100	make a table with					
r: 100	spaces unfortunately cannot						
c: 100	be represented visually here,	Function: Constructor_MAX					
to_win: 25	but that is what is created.						

## boolean checkIfFree(int c)

Input: State: (number to win = 3	Output: false State: state of board is	Reason: This test case is unique
X X	unchanged	because it tests checkIfFree on a full column and is designed to returned false
X		Function: checkIfFree_column_full
X C = 0		

# boolean checkIfFree(int c) Input: Output: tr

Input: State: (number to win = 3)	Output: true State: state of board is unchanged	Reason: This test case is unique because it tests checkifFree on
		an empty column, and therefore should return true
		Function: checkIfFree_column_empty
C = 0		

#### boolean checkIfFree(int c)

Input: State:	(number t	o win =	3)	Output: true State: state of board is	Reason: This test case is unique because
X	X			unchanged	it tests checklfFree on a partially full column rather than a full column or empty one
X					Function: checkIfFree_column_partially_full
C = 0					

## boolean checkHorizWin(int r, int c, char p)

Inpu		ber to	win = 3)	Output: true State: state of board is	Reason: This test case is unique
				unchanged	because it tests checkHorizWin from the beginning of a valid horizontal as opposed to the middle or end
X	Х	X			Function: checkHorizWin_row0_beginning
R = 0 c = 0 p = '	)				

#### boolean checkHorizWin(int r, int c, char p)

Input: State: (nur	nput: tate: (number to win = 3)			Output: true State: state of board is	Reason: This test case is unique
				unchanged	because it tests checkHorizWin from the end of a valid horizontal as opposed to the middle or beginning  Function:
R = 0 c = 4 p = 'X'	X	X	X		checkHorizWin_row0_end

#### boolean checkHorizWin(int r, int c, char p)

Input: State: (number to win = 3)			3)	Output: true State: state of board is	Reason: This test case is unique
				unchanged	because it tests checkHorizWin from the middle of a valid horizontal as opposed to the end or beginning  Function:
X	X	X			checkHorizWin_row0_middle
R = 0					
c = 2 p = 'X'					

	Input: State: (number to win = 3)			3)	Output: false State: state of board is	Reason: This test case is unique
					unchanged	because it tests checkHorizWin from the beginning of an invalid horizontal as opposed to the middle or end
X	X	0	0			Function: checkHorizWin_row0_false_beg
c = 0	R = 0 c = 0 p = X				inning	illining
boole	an che	ckHori	zWin(i	nt r, int	c, char p)	
Input State		nber to	win =	3)	Output: false State: state of board is	Reason: This test case is unique
					unchanged	because it tests checkHorizWin from the end of an invalid horizontal as opposed to the middle or beginning
	X	Y	0	0		Function: checkHorizWin row0 false end
X X O O C   X   X   X   X   X   X   X   X   X						5.125.kt.131.121111_13113_14136_e114

# boolean checkVertWin(int r, int c, char p)

Input: State: (number	r to win = 3)	Output: true State: state of board is	Reason: This test case is unique
		unchanged	because it tests checkVertWin on a column with only the winning characters inside, as
X			opposed to some below it
X			Function: checkVertWin_top
X			
R = 2			
c = 0			
p = X			

# boolean checkVertWin(int r, int c, char p)

Input: State:	: (number to	win = 3)	Output: true State: state of board is	Reason: This test case is unique
X			unchanged	because it tests checkVertWin on a winning column with both
X				winning and nonwinning
X				characters inside, as opposed
0				to exclusively winning chars
0				Function:
R = 2		'		checkVertWin_top_spots_below
c = 0				
p = X				

#### boolean checkVertWin(int r, int c, char p)

Input: State: (number to win = 3)		Reason: This test case is unique because it tests checkVertWin on a non winning column with different characters inside from an empty spot above the highest char in the column, as opposed to exclusively one type of char
X X O	unchanged	
O R = 4 c = 0 p = X		Function: checkVertWin_false_empty_spots_below

## boolean checkVertWin(int r, int c, char p)

Input: State:	(numb	er to v	win = 3	3)	Output: false State: state of board is	Reason: This test case is unique because it tests
X					unchanged	checkVertWin on a non winning column with different characters inside, as opposed to exclusively one type of char
x						
0						Function: checkVertWin_false_top_spots_below
0						
R = 3 c = 0 p = X	1					

## boolean checkVertWin(int r, int c, char p)

Input: State: (number to win = 3)		Reason: This test case is unique because it tests checkVertWin on an empty board, and is trying to make sure it does not check indices that do not exist, causing null ptr exception
	unchanged	
		Function: checkVertWin_false_empty_board
R = 0 c = 0		
p = X		

## boolean checkDiagWin(int r, int c, char p)

Input: State: (number to win = 3)  R = 0 c = 0	Output: false State: state of board is unchanged	Reason: This test case is unique because it tests checkDiagWin on an empty board, and is trying to make sure it does not check indices that do not exist, causing null ptr exception Function: checkDiagWin_false_empty_board
c = 0 p = X		

## boolean checkDiagWin(int r, int c, char p)

Input State		nber to	win = 3)	unchanged	Reason: This test case is unique because it tests checkDiagWin on a diagonal where the left is the bottom and the right side is the top, as opposed to the opposite way of right bottom left top
		X			
	Х	Х			Function:
X	X	X			checkDiagWin_left_bottom_right_top
R = 2 $c = 2$ $p = 3$				-	

#### boolean checkDiagWin(int r, int c, char p)

Input:				Output: true	Reason:
X X	X X	x	win = 3)	State: state of board is unchanged	This test case is unique because it tests checkDiagWin on a diagonal where the left is the top and the right side is the bottom, as opposed to the opposite way of right top left bottom  Function: checkDiagWin_left_top_right_bottom

## boolean checkDiagWin(int r, int c, char p)

Inpu	t:				Output: true	Reason:
State	e: (nur	nber to	o win =	= 3)	State: state of board is	This test case is unique because it tests
				Χ	unchanged	checkDiagWin on a diagonal where the
			Х	Х		winning characters have a pattern of left is the bottom and the right side is
		Х	Х	Х		the top with non winning characters
0	0	0	0	0		below, as opposed to the opposite way of right bottom left top
0	О	0	0	0		Finalism
R = 4 c = 4 p = X				·	_	Function: checkDiagWin_left_bottom_right_top_sp ots_under

#### boolean checkDiagWin(int r, int c, char p)

Input:					Output: true	Reason:
State	: (num	ber to	win =	3)	State: state of board is	This test case is unique because it tests
X					unchanged	checkDiagWin on a diagonal where the
X	Х					winning characters have a pattern of left is the top and the right side is the
Х	Χ	Х				bottom with non winning characters below, as opposed to the opposite way
0	0	0	0	0		of right bottom left top
0	О	0	0	0		Function:
R = 4 $c = 0$						checkDiagWin_left_top_right_bottom_sp ots under
p = X						ots_ander

## boolean checkDiagWin(int r, int c, char p)

Input State		oer to	win = 3)		State: state of board is	Reason: This test case is unique because it tests checkDiagWin on a non winning diagonal where the left is the bottom and the right side is the top, as opposed to the opposite way of right bottom left top
	X					
X R = 0 c = 0 p = X						Function: checkDiagWin_left_bottom_right_top_in sufficient_chars

#### boolean checkDiagWin(int r, int c, char p)

Input: State: (number to win = 3)		Reason: This test case is unique because it tests checkDiagWin on a non winning diagonal where the left is the top and the right side is the bottom, as opposed to the opposite way of right top left bottom
X	unchanged	
R = 1 c = 1 p = X		Function: checkDiagWin_false_left_top_right_bott om_insufficient_chars

## boolean checkDiagWin(int r, int c, char p)

Input State		nber to	win =	= 3)	Output: false State: state of board is	Reason: This test case is unique because it tests
0	0	0	0	0	unchanged	checkDiagWin on a board that is
Х	Х	Х	Х	Х		alternating, allowing it to test all diagonals negatively, making sure that
0	0	0	0	0		there are no false positives for these
X	Х	Х	Х	Х		cases
0	0	0	0	0		Function:
R = 2						checkDiagWin_full_alternating_board
c = 2						
p = 0	)					

## boolean checkTie()

Input	t:				Output: true	Reason:
State	State: (number to win = 3)				State: state of board is	This test case is unique because it tests
X	X	X	X	X	unchanged	checkTie on a board that is full as
Х	Х	Х	Х	Х		opposed to empty or partially full
Х	Х	Х	Х	Х		Function: checkTie_full_board
Х	Х	Х	Х	Х		
Х	Х	Х	Х	Х		

## boolean checkTie()

Input: State: (number to win = 3)	State: state of board is unchanged This te	Reason: This test case is unique because it tests checkTie on an empty board as opposed to full or partially full
		Function: checkTie_empty_board

#### boolean checkTie()

Input State		ber to	win = 3)	unchanged	Reason: This test case is unique because it tests checkTie on a board where only some
X	Х	Х			
X	Х	Х			of the columns are full as opposed to some of the rows full or entire board
X	Х	Х			empty or full
X	Х	Х			Function: checkTie some columns full
Х	Х	Х			

#### boolean checkTie()

Input State		ber to	win =	3)		
0	0	0	0	0		checkTie on a full, alternating character
X	Х	Х	Х	Χ		board as opposed to a non-alternating full board, partially full, or empty board
0	0	0	0	0		Function
X	Х	Х	Х	Х		Function: checkTie full alternating board
0	О	0	О	0		

## char whatsAtPos(int r, int c)

	(	-,		_
Input:			Output: ' '	Reason:
State: (numb	er to win =	3)	State: state of board is	This test case is unique because it tests
			unchanged	WhatsAtPos on an empty space on an empty board as opposed to full or partially full
				Function: WhatsAtPos_empty_space_empty_board
R = 0	'			
c = 0				

# char whatsAtPos(int r, int c)

Input	:				Output: ' '	Reason:
State	State: (number to win $= 3$ )				State: state of board is	This test case is unique because it tests
					unchanged	WhatsAtPos on an empty space on an partially full board as opposed to full or
						empty one
						Function: WhatsAtPos empty space part full boa
X	X	X	Χ	Χ		rd
R = 1 c = 0						

#### char whatsAtPos(int r, int c)

that whatsAtros(incl) incl)								
Input	:				Output: ' '	Reason:		
State	State: (number to win = 3)				State: state of board is	This test case is unique because it tests		
Х	Х	Х	Х		unchanged	WhatsAtPos on the only empty space on an nearly filled board as opposed to		
X	Х	Х	X	Х		full or empty board		
X	X	X	X	Χ		Function:		
Χ	X	X	X	Χ	11	WhatsAtPos_empty_space_almost_full_b		
X	X	X	X	Χ		oard		
R = 4								
c = 4								

## char whatsAtPos(int r, int c)

Input: State: (numb	per to win = :	3)		Reason: This test case is unique because it tests WhatsAtPos on the only non-empty space on the board as opposed to a completely full or empty board
			unchanged	
				Function: WhatsAtPos_only_spot_on_board
	x			
R = 0 c = 3				

## char whatsAtPos(int r, int c)

Input: State	: : (numl	oer to	win =	: 3)	Output: 'X' State: state of board is unchanged	Reason: This test case is unique because it tests WhatsAtPos on a different char from the rest of its row as opposed to the same one as the rest, making sure it is picking the right one on the row
0	0	0	Х	0		Function: WhatsAtPos_diff_char_than_rest_of_row
R = 0 $c = 3$		1	1			

## char whatsAtPos(int r, int c)

Input: State:		oer to	win = :	3)	Output: 'X' State: state of board is	Reason: This test case is unique because it tests
О	0	0	0	0	unchanged	WhatsAtPos on a different char from the
0	0	0	0	0		rest of the entire board as opposed to the same one as the rest, making sure
0	0	Х	0	0		it is picking the right one on the board
0	0	0	0	О		Function:
0	0	0	0	0		WhatsAtPos_diff_char_than_rest_of_boar
R = 2 c = 2						d
C = 2						

#### char whatsAtPos(int r, int c)

Input: State: (number to win = 3)	Output: 'X' State: state of board is	Reason: This test case is unique because it tests
	unchanged	WhatsAtPos when there are different characters above and below it, making sure the function is picking the right
0		character
X		Function:
0		WhatsAtPos_spots_taken_above_and_be low
$ \begin{vmatrix} R = 1 \\ c = 0 \end{vmatrix} $		IOW

# void placeToken(char p, int c)

Input: State: (number to win = 3)	Output: State:	Reason: This test case is unique because it
		tests if placeToken can put a character on an empty board as opposed to a partially full one
		Function: placeToken_empty_board
p = X c = 0	X	

## void placeToken(char p, int c)

Input: State: (number to win = 3)	Output: State:	Reason: This test case is unique because it
		tests if placeToken can put a char on a column that is partially full, as opposed to empty or 1 away from full
	X	Function:
0	0	placeToken_column_part_full
P = 'X' $c = 0$		

#### void placeToken(char p, int c)

Input:	Output:	Reason:
State: (number to win = 3)		This test case is unique because it tests if placeToken can put a char
	X	on a column that would completely
0	0	fill the column as opposed to a
0	0	column with nothing or only 1 or 2 chars inside
0	0	Chars inside
0	0	Function: placeToken_to_fill_column
P = 'X'		
c = 0		

void placeToken(char p, int c)

Input: State: (number to win = 3)				Outp					Reason: This test case is unique because it	
										tests if placeToken can put a char on a column that would completel fill the row as opposed to a row with nothing or only 1 or 2 chars inside
О	О	0	О		О	0	0	0	X	Function: placeToken_to_fill_row
P = c = -								·		

void placeToken(char p, int c)

		rente	nar p,	1110 07	Ot.	~ · · + ·				Danasa	
Input:			Outp					Reason:			
State	State: (number to win = 3)			Stat	<del>е</del> .				This test case is unique because it		
0	0	0	0		0	0	0	0	X	tests if placeToken can put a char on a column that would completely	
0	0	0	0	0	О	0	0	0	0	fill the board as opposed to a boa	
0	0	0	0	0	О	0	0	0	0	with nothing or only 1 or 2 chars inside	
0	0	0	0	0	О	0	0	0	0	Inside	
0	0	0	0	0	О	0	0	0	0	Function: placeToken_to_fill_board	
P = '	X'	•	•					,			
c = 4	c = 4										