Parthiv:

Test Cases:

Constructor: GameBoard(int row, int column, int numTokens) Test Case 1:

Input:	**Output:**	**Reason:**

State:
int row = 3
int column = 3
Int numTokens = 3
	getNumRows = row;
	getNumColumns = column;
	getNumToWin = numTokens;
	Expected output of board is empty
		The following test case is unique because the constructor is passed on with the minimum values for, number of rows, columns and tokens required for the game.
		.
		Function Name:
		testConstructor_min_values

Test Case 2:

Input:	**Output:**	**Reason:**

State:
int row = 100
int column = 100
Int numTokens = 25
	getNumRows = row;
	getNumColumns = column;
	getNumToWin = numTokens;
	Expected output of board is empty
		The following test case is unique because constructor is passed on with the maximum values for number of rows, columns and tokens required for the game.
		Function Name:
		testConstructor_max_values

Test Case 3:

Input:	**Output:**	**Reason:**

State:
int row = 27
int column = 55
Int numTokens = 15
	getNumRows = row;
	getNumColumns = column;
	getNumToWin = numTokens;
	
	Expected output of board is empty
		The following test case is unique because the constructor is passed on with all the random values for setting up the game board.
		
		Function Name:
		testConstructor_random_values

CONNOR:
void checkIfFreeFalse()

<table>
<tr><th>Input:</th><th>Output:</th><th>Reason:</th></tr>
<tr>
<td>
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
</table>
Int col = 3
</td>
<td>checkIfFreeFalse= false

State of the board is unchanged</td>
<td>This test case test to make sure we the third colum is not free

<b>Function Name:</b>
void checkIfFreeFalse()</td>
</tr>
</table>

void checkIfFreeTrue()

<table>
<tr>
<td><b>Input:</b>
<table>
<tr><td></td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td></td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td></td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td></td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td></td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
</table>
Int col = 0</td>
<td><b>Output:</b>
checkIfFreeFalse= True
State of the board is unchanged</td>
<td><b>Reason:</b>
This test case test to make sure we the First colum is not free
<b>Function Name:</b>
void checkIfFreeTrue()</td>
</tr>
</table>

void checkIfFree99True()

<table>
<tr>
<td><b>Input:</b>
<table>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td></td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
</table>
Int col = 5</td>
<td><b>Output:</b>
checkIfFreeFalse= True
State of the board is unchanged</td>
<td><b>Reason:</b>
This test case test to make sure we the First colum is not free
<b>Function Name:</b>
void checkIfFreeTrue()</td>
</tr>
</table>

void checkVertWinBottomTrue()

<table>
<tr>
<td><b>Input:</b>
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
</table>
</td>
<td><b>Output:</b>
checkforVertWin = true
State of the board is unchanged</td>
<td><b>Reason:</b>
This test case test the limit of the bottom and the far left
<b>Function Name:</b>
void checkVertWinBottomTrue()</td>
</tr>
</table>

X X X X X X X X X X X O O O O O O O O O O O X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X pos.getRow = 11 pos.getCol = 0 P = ‘X’		

void checkVertWinBottomFlase()

Input:	**Output:**	**Reason:**
X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X O O O O O O O O O O O X X X X X X X X X X X X X X X X X X X X X X pos.getRow = 0 pos.getCol = 10 P = ‘X’	checkforVertWin = false State of the board is unchanged	This test case test the limit of the bottom and the far right **Function Name:** void checkVertWinBottomFlase()

void checkVertWinTopFalse()

<table>
<tr>
<td><b>Input:</b>
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
</table>

pos.getRow = 10
pos.getCol = 0
P = ‘X’</td>
<td><b>Output:</b>

checkforVertWin = false

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case test the limit of the top and the far right

<b>Function Name:</b>
void checkVertWinBottomFlase()</td>
</tr>
</table>

void checkVertWinTopTrue()

<table>
<tr>
<td><b>Input:</b>
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
</table></td>
<td><b>Output:</b>

checkforVertWin = true

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case test the limit of the top and the far right

<b>Function Name:</b>
void checkVertWinTopTrue()</td>
</tr>
</table>

X X X X X X X X X X X O O O O O O O O O O O X X X X X X X X X X X X X X X X X X X X X X pos.getRow = 10 pos.getCol = 10 P = ‘X’		

void checkHorzWinBotLeftTrue()

Input:	**Output:**	**Reason:**
X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X X X X X X pos.getRow = 10 pos.getCol = 0 P = ‘X’	checkforHoriztWin = true State of the board is unchanged	This test case test the limit of the top and the far left **Function Name:** void checkHorzWinBotLeftTrue()

void checkHorzWinTopRightTrue()

Input:	**Output:**	**Reason:**
	checkforHoriztWin = True	This test case test the limit of the top and the far right

<table>
<tr>
<td>
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
</table>

pos.getRow = 0
pos.getCol = 10
P = ‘X’
</td>
<td>State of the board is unchanged</td>
<td><b>Function Name:</b>
void
checkHorzWinTopRightTrue()</td>
</tr>
</table>

void checkHorzWinBotRightFalse()

<table>
<tr>
<td>
<b>Input:</b>
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>O</td><td>X</td><td>X</td><td>X</td></tr>
</table>
</td>
<td><b>Output:</b>
checkforHoriztWin = false
State of the board is unchanged</td>
<td><b>Reason:</b>
This test case test the limit of the Bottom and the far right
<b>Function Name:</b>
void
checkHorzWinBotRightFalse( )</td>
</tr>
</table>

X X X X X X X O X X X
X X X X X X X O X X X
 	
pos.getRow = 0
pos.getCol = 10
P = ‘X’		

void checkHorzWinTopLeftTrue()

Input:	**Output:**	**Reason:**
X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X X X X X X X X O X X X pos.getRow = 10 pos.getCol = 2 P = ‘X’	checkforHoriztWin = true State of the board is unchanged	This test case test the limit of the Bottom and the far left **Function Name:** void checkHorzWinTopLeftTrue()

Table by Connor & Parthiv, Output and reason by Parthiv
boolean checkDiagtWinLeftToRightBotTrue()

Input: **State: (number to win = 4)**	**Output:** checkDiagWin = true	**Reason:**
O O O O O	State of the board is	This test case is distinct because it checks if checkDiagWin is true when

<table>
<tr>
<td>
<table>
<tr><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
</table>
pos.getRow = 0
pos.getCol = 0
P = ‘X’
</td>
<td>unchanged</td>
<td>there is a diagonal chain from left bottom to top right

<b>Function Name:</b>
void
checkDiagtWinLeftToRightBot
True</td>
</tr>
</table>

test_checkDiagWin_win_middle_left_to_middle_to_right()

<table>
<tr>
<td>
<b>Input:</b>
<b>State: (number to win = 4)</b>
<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>X</td><td></td><td></td><td></td></tr>
<tr><td></td><td>O</td><td>X</td><td></td><td></td></tr>
<tr><td></td><td>O</td><td>O</td><td>X</td><td></td></tr>
</table>
pos.getRow = 2
pos.getCol = 2
P = ‘X’
</td>
<td><b>Output:</b>
checkDiagWin = true

State of the board is unchanged</td>
<td><b>Reason:</b>
The following test case is unique because the last placed token in the middle resulted in 3 consecutives X’s, but the function has to count from top left to middle and also from middle to right bottom.
<b>Function Name:</b>
void
checkDiagtWinLeftToRightFal
se</td>
</tr>
</table>

checkDiagtWinLeftToRightTopRightTrue()

<table>
<tr>
<td>
<b>Input:</b>
<b>State: (number to win = 4)</b>
<table>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td></tr>
</table>
</td>
<td><b>Output:</b>
checkDiagWin = true

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case is distinct because it checks if checkDiagWin is true when there is a chain from left to top right

<b>Function Name:</b></td>
</tr>
</table>

<table>
<tr>
<td>
<table>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
</table>
pos.getRow = 0
pos.getCol = 0
P = ‘X’
</td>
<td></td>
<td>void checkDiagtWinLeftToRightTopRightTrue</td>
</tr>
</table>

checkDiagtWinRightToLeftBotRightTrue()

<table>
<tr>
<td>
<b>Input:</b>
<b>State: (number to win = 4)</b>
<table>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td></tr>
</table>
pos.getRow = 0
pos.getCol = 0
P = ‘X’
</td>
<td><b>Output:</b>
checkDiagWin = true

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case is distinct because it checks if checkDiagWin is true when there is a chain from bottom right to top left

<b>Function Name:</b>
void checkDiagtWinRightToLeftBotRightTrue</td>
</tr>
</table>

void checkDiagtWinRightToLeftTopLeftTrue()

<table>
<tr>
<td>
<b>Input:</b>
<b>State: (number to win = 4)</b>
<table>
<tr><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td><td>O</td></tr>
</table>
</td>
<td><b>Output:</b>
checkDiagWin = true

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case is distinct because it checks if checkDiagWin is true when there is a chain from right to top left

<b>Function Name:</b></td>
</tr>
</table>

O O O O O O O
O O O O O O O
O O O O O O O
pos.getRow = 5
pos.getCol = 1
P = ‘X’
		void checkDiagtWinRightToLeftTopLeftTrue

void checkDiagtWinRightToLeftFalse()

Input: **State: (number to win = 4)** X O O O O O O O X O O O O O O O X O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O pos.getRow = 5 pos.getCol = 1 P = ‘X’	**Output:** checkDiagWin = false State of the board is unchanged	**Reason:** This test case is distinct because it checks if checkDiagWin is false when there is a chain from Left to right **Function Name:** void checkDiagtWinRightToLeftFalse

void checkDiagtWinRightToLeftBotLeftTrue()

Input: **State: (number to win = 4)** X O O O O O O O X O O O O O O O X O O O O O O O X O O O	**Output:** checkDiagWin = true State of the board is unchanged	**Reason:** This test case is distinct because it checks if checkDiagWin is true when there is a chain from Left to bottom right **Function Name:**

<table>
<tr>
<td>
<table>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td><td>O</td></tr>
<tr><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>O</td><td>X</td></tr>
</table>
pos.getRow = 0
pos.getCol = 6
P = ‘X’
</td>
<td></td>
<td>void checkDiagtWinRightToLeftBotLeftTrue</td>
</tr>
</table>

Prahalad:

boolean isPlayerAtPos(Boardposition pos, char player)

<table>
<tr>
<td>
<b>Input:</b>
<b>State: (number to win = 3)</b>
<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
pos.getRow = 0
pos.getCol = 0
P = ‘X’
</td>
<td><b>Output:</b>
isPlayerAtPos = false

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case is distinct because it checks isPlayerAtPos on an empty space

<b>Function Name:</b>
void isPlayerAtPos_false_empty_spot_empty_board</td>
</tr>
</table>

boolean isPlayerAtPos(Boardposition pos, char player)

<table>
<tr>
<td>
<b>Input:</b>
<b>State: (number to win = 3)</b>
<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>O</td></tr>
</table>
</td>
<td><b>Output:</b>
isPlayerAtPos = true

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case is distinct because it checks isPlayerAtPos on one spot with a token on a board that is empty on all other spots

<b>Function Name:</b>
void isPlayerAtPos_one_char_on_board</td>
</tr>
</table>

<table>
<tr><td>pos.getRow = 0
pos.getCol = 4
P = ‘O’</td><td></td><td></td></tr>
</table>

boolean isPlayerAtPos(Boardposition pos, char player)

<table>
<tr>
<td><b>Input:</b>
<b>State: (number to win = 3)</b>
<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
</table>
pos.getRow = 0
pos.getCol = 1
P = ‘X’</td>
<td><b>Output:</b>
isPlayerAtPos = true

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case is distinct because it checks isPlayerAtPos on a character on a filled row

<b>Function Name:</b>
void isPlayerAtPos_one_filled_row</td>
</tr>
</table>

boolean isPlayerAtPos(Boardposition pos, char player)

<table>
<tr>
<td><b>Input:</b>
<b>State: (number to win = 3)</b>
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
</table>
pos.getRow = 4
pos.getCol = 4
P = ‘O’</td>
<td><b>Output:</b>
isPlayerAtPos = false

State of the board is unchanged</td>
<td><b>Reason:</b>

This test case is distinct because it checks isPlayerAtPos on an empty space with an otherwise full board

<b>Function Name:</b>
void isPlayerAtPos_false_almost_full_board_empty_space</td>
</tr>
</table>

boolean isPlayerAtPos(Boardposition pos, char player)

<table>
<tr>
<td><b>Input:</b>
<b>State: (number to win = 3)</b>
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td></tr>
</table>
pos.getRow = 0
pos.getCol = 0
P = ‘X’</td>
<td><b>Output:</b>
isPlayerAtPos = true
State of the board is unchanged</td>
<td><b>Reason:</b>
This test case is distinct because it checks isPlayerAtPos on a character in a completely filled board
<b>Function Name:</b>
void isPlayerAtPos_full_board</td>
</tr>
</table>

void dropToken(char p, int c)

<table>
<tr>
<td><b>Input:</b>
<b>State: (number to win = 3)</b>
<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
p = ‘X’
c = 0</td>
<td><b>Output:</b>
<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>X</td><td></td><td></td><td></td><td></td></tr>
</table>
</td>
<td><b>Reason:</b>
This test case is distinct because it checks if a token was successfully placed on an empty board
<b>Function Name:</b>
void dropToken_on_empty_board</td>
</tr>
</table>

void dropToken(char p, int c)

<table>
<tr>
<td><b>Input:</b>
<b>State: (number to win = 3)</b></td>
<td><b>Output:</b></td>
<td><b>Reason:</b>
This test case is distinct because it checks if a token</td>
</tr>
</table>

<table>
<tr>
<td><table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>O</b></td><td></td><td></td><td></td><td></td></tr>
</table>
p = ‘X’
c= 0</td>
<td><table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>O</b></td><td></td><td></td><td></td><td></td></tr>
</table></td>
<td>was successfully placed in a partly filled column
<b>Function Name:</b>
void dropToken_in_partly_filled_column</td>
</tr>
</table>

void dropToken(char p, int c)

<table>
<tr>
<td><b>Input:</b>
<b>State: (number to win = 3)</b>
<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td></td><td></td></tr>
</table>
p = ‘X’
c = 4</td>
<td><b>Output:</b>
<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td></td><td><b>X</b></td></tr>
</table></td>
<td><b>Reason:</b>
This test case is distinct because it checks if a token was successfully placed in a partly filled row
<b>Function Name:</b>
void dropToken_in_partly_filled_row</td>
</tr>
</table>

void dropToken(char p, int c)

<table>
<tr>
<td><b>Input:</b>
<b>State: (number to win = 3)</b>
<table>
<tr><td><b>X</b></td><td><b>X</b></td><td><b>O</b></td><td><b>X</b></td><td></td></tr>
<tr><td><b>X</b></td><td><b>X</b></td><td><b>O</b></td><td><b>X</b></td><td><b>O</b></td></tr>
<tr><td><b>X</b></td><td><b>X</b></td><td><b>O</b></td><td><b>X</b></td><td><b>O</b></td></tr>
</table></td>
<td><b>Output:</b>
<table>
<tr><td><b>X</b></td><td><b>X</b></td><td><b>O</b></td><td><b>X</b></td><td><b>O</b></td></tr>
<tr><td><b>X</b></td><td><b>X</b></td><td><b>O</b></td><td><b>X</b></td><td><b>O</b></td></tr>
<tr><td><b>X</b></td><td><b>X</b></td><td><b>O</b></td><td><b>X</b></td><td><b>O</b></td></tr>
</table></td>
<td><b>Reason:</b>
This test case is distinct because it checks if a board can be filled with tokens
<b>Function Name:</b>
void dropToken_to_fill_board</td>
</tr>
</table>

<table>
<tr>
<td><table><tr><td>X</td><td>X</td><td>O</td><td>X</td><td>O</td></tr><tr><td>X</td><td>X</td><td>O</td><td>X</td><td>O</td></tr></table>p = ‘O’
c = 4</td>
<td><table><tr><td>X</td><td>X</td><td>O</td><td>X</td><td>O</td></tr><tr><td>X</td><td>X</td><td>O</td><td>X</td><td>O</td></tr></table></td>
<td></td>
</tr>
</table>

Sean Farrell Next 9 functions

void checkWhatsAtPos_checks_true

<table>
<tr>
<td><b>Input: (needs to change)</b>
State (number to win == 6)
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>P = ‘X’</td>
<td><b>Output: (needs to change)</b>

checkWhatAtPos = ‘X’.
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>X</td></tr>
</table></td>
<td><b>Reason:</b> This test case is distinct because it makes shure the returning the value of the board or the most bottom right position which is populated by ‘X’’.
<b>Function Name:</b>
void checkWhatsAtPos_checks_true</td>
</tr>
</table>

void checkWhatsAtPos_checks_False

<table>
<tr>
<td><b>Input: (needs to change)</b>
State: (number to win == 6)
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

P= ‘X’</td>
<td><b>Output: (needs to change)</b>

checkWhatAtPos != ‘O’.
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table></td>
<td><b>Reason:</b> This test case is distinct because it makes shure that whatAtPos is returning the correct value of the populated bottom left which is ‘X’ of the board. Its not returning invalid information which would be ‘O’.
<b>Function Name:</b>
void checkWhatsAtPos_checks_False</td>
</tr>
</table>

void checkWhatsAtPos_top_of_inserted_column

<table>
<tr>
<td><b>Input: (needs to change)</b>
State (number to win == 6)
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table></td>
<td><b>Output: (needs to change)</b>
checkWhatAtPos = ‘x’.
<table>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table></td>
<td><b>Reason:</b> This test case is distinct because it checks the leftmost populated column’s contents top element is the same contents in the gameboard. In this scenario it is ‘X which whatAtPos returns ‘X’ due to populating the column complete with ‘X’.
<b>Function Name:</b>
void checkWhatsAtPos_top_of_inserted_column</td>
</tr>
</table>

<table>
<tr>
<td><table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
P = ‘X’</td>
<td><table>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>X</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table></td>
<td></td>
</tr>
</table>

void checkWhatsAtPos_empty_position

<table>
<tr>
<td><b>Input: (needs to change)</b>
State (number to win == 6)
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
P = ‘X’</td>
<td><b>Output: (needs to change)</b>
checkWhatAtPos = ‘ ‘.
State of the gameboard is unchanged</td>
<td><b>Reason:</b> This test case is distinct it makes shure when a game board is constructed before the game starts,The top left conor will always be empty at that position. Checks to see if whatAPos returns ‘ ‘ which is empty for that space.
<b>Function Name:</b>
void checkWhatsAtPos_empty_position</td>
</tr>
</table>

void checksWhatsAtPos_rightmost_position_inserted_row

Input: (needs to change) State (number to win == 6)	**Output: (needs to change)** checkWhatAtPos = ‘O’.	**Reason:** This test case is distinct because it checks the rightmost populated piece in the lowest row. It

<table>
<tr>
<td>
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
P = ‘O’
</td>
<td>
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td><td><b>O</b></td></tr>
</table>
</td>
<td>finds and returns ‘O’ due to populating that space with a row entirely of ‘O’ pieces.
<b>Function Name:</b>
void checkWhatsAtPos_top_of_inserted_column</td>
</tr>
</table>

void checkTie_full_board_no_wins

<table>
<tr>
<td><b>Input: (needs to change)</b>
State (number to win == 3)
<table>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>
P = ‘X’
</td>
<td><b>Output:</b>
checkTie = true.
<table>
<tr><td><b>O</b></td><td><b>O</b></td><td><b>X</b></td></tr>
<tr><td><b>X</b></td><td><b>X</b></td><td><b>O</b></td></tr>
<tr><td><b>O</b></td><td><b>X</b></td><td><b>X</b></td></tr>
</table>
</td>
<td><b>Reason:</b> This test case is distinct because it populates a game board with pieces with a scenario with no winners.
<b>Function Name:</b>
void checkTie_full_board_no_wins</td>
</tr>
</table>

void checkTie_partially_filled_board_with_win

<table>
<tr>
<td><b>Input: (needs to change)</b>
State (number to win == 3)
<table>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>
</td>
<td><b>Output:</b>
checkTie = false.
<table>
<tr><td></td><td></td><td><b>O</b></td></tr>
<tr><td><b>X</b></td><td><b>O</b></td><td><b>O</b></td></tr>
</table>
</td>
<td><b>Reason:</b> This test case is distinct because it has a finished game where ‘O’ wins making shure the checktie method doesnt mistake a win as a tie in a not fully populated gameboard.
<b>Function Name:</b></td>
</tr>
</table>

<table>
<tr>
<td><table><tr><td></td><td></td><td></td></tr></table>
P = ‘X’</td>
<td><table><tr><td><b>O</b></td><td><b>X</b></td><td><b>X</b></td></tr></table></td>
<td>Void checkTie_partially_filled_board_with_win</td>
</tr>
</table>

void checkTie_empty_board

<table>
<tr>
<td><b>Input: (needs to change)</b>
State (number to win == 3)
<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>
P = ‘X’</td>
<td><b>Output:</b>
checkTie = false.

State of the gameboard is unchanged</td>
<td><b>Reason:</b> This test case is distinct because when a gameboard is constructed before the game has started. CheckTie should be false due to the game has not even started yet. You cant tie a game if you havent started the game.
<b>Function Name:</b>
void checkTie_empty_board</td>
</tr>
</table>

checkTie_partaily_filled_board_with_no_win

<table>
<tr>
<td><b>Input: (needs to change)</b>
State (number to win == 3)
<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>
P = ‘X’</td>
<td>checkTie = false.
<table><tr><td><b>O</b></td><td></td><td></td></tr><tr><td><b>X</b></td><td><b>O</b></td><td><b>O</b></td></tr><tr><td><b>O</b></td><td><b>X</b></td><td><b>X</b></td></tr></table></td>
<td><b>Reason:</b> This test case is distinct because when a game is currently being played and the board is not yet completely full the checktie function should return false.
<b>Function Name:</b>
void checkTie_empty_board</td>
</tr>
</table>