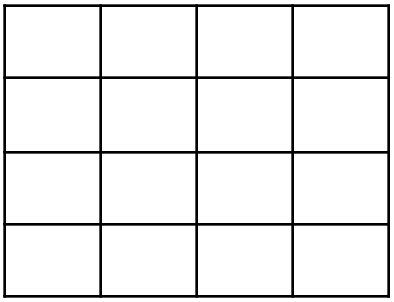
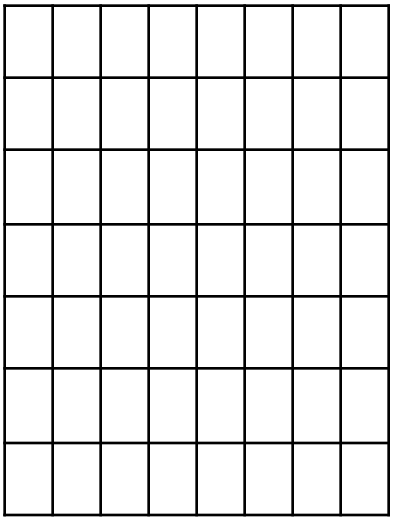
Constructor Test cases:
constructor_test_1()

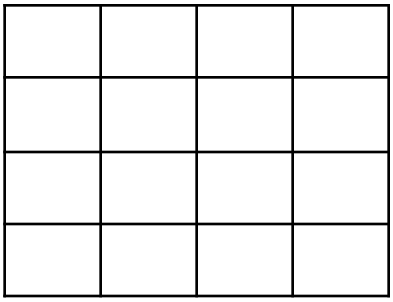
Input:	Output: Create instance State:(num to win 25)  100x100 board above All values on board are empty spaces	Reason: test case because to see if max size table is possible and does not crash Function name: constructor_test_1

constructor_test_equalRowColandNumsToWin()

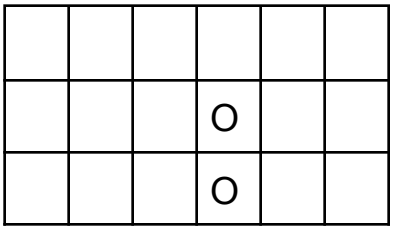
Input:	Output Create instance State: numstoWin = 8 row= 8 Column = 8 	Reason: test to see if all variables are the same val that the program can handle the special case Function Name constructor_test_equalRowColandNumsToWin

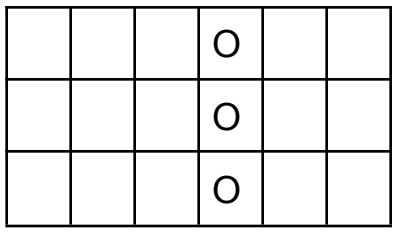
		

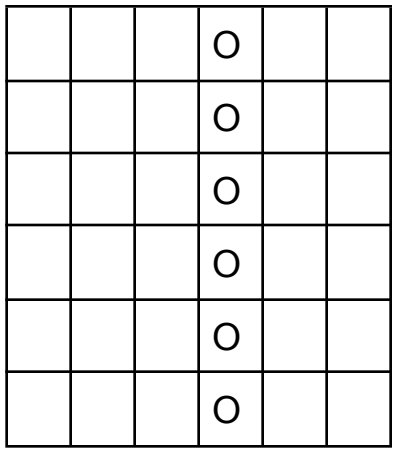
Constructor_test_doubleDigetSize()

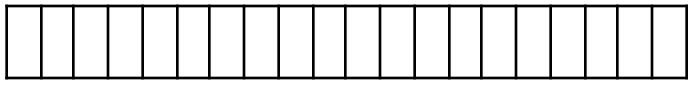
input	Output State: Creates an instance Of a 40x32 board Row =40 Column = 32 numToWin = 20 	Reason creates a routine case in the middle of the max and min values. FunctionName: Constructor_test_doubleDigetSize

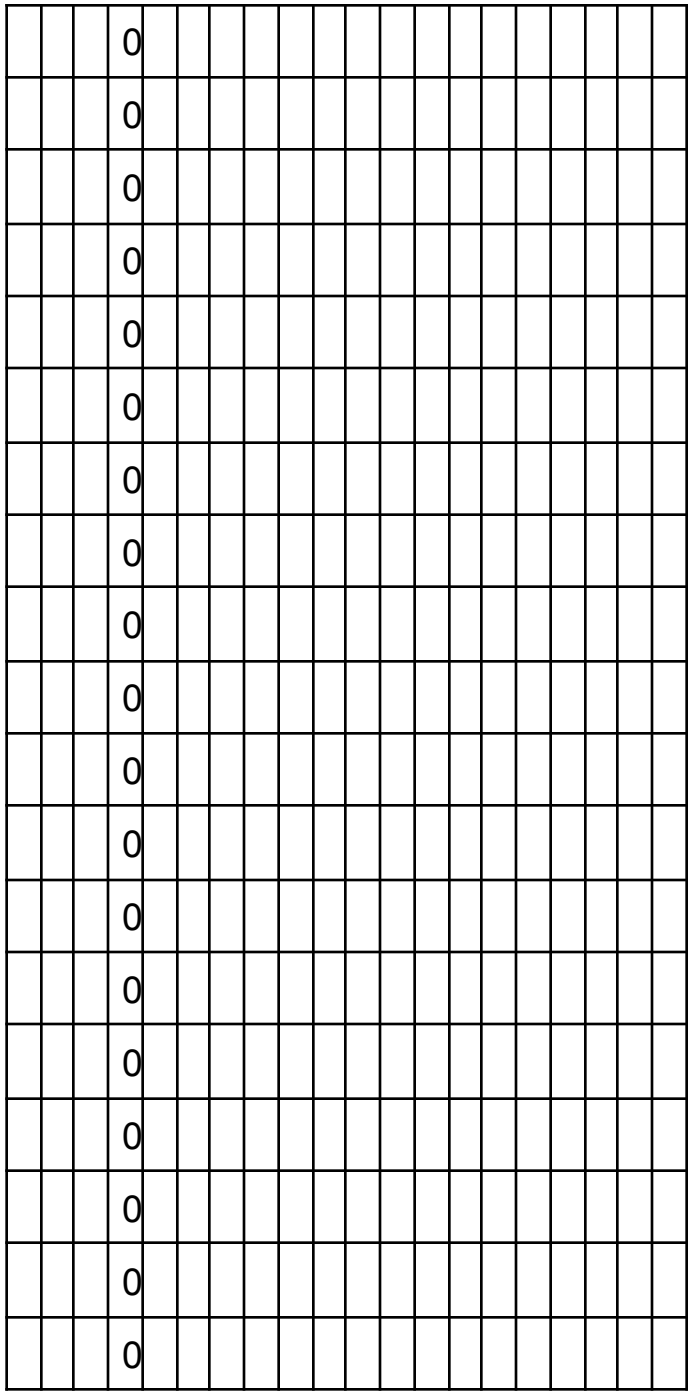
checkIfFree Test cases:

Input State: numToWin = 3 Row = 6 Col = 6 	Output checkifFree: returns true State of board unchanged	Reason: routine case Function name checkIfFree_test_1

Input State: numToWin =3 Row =6 Column =6 	Output: CheckifFree: returns false State of board unchanged	Reason: checking to make sure the method knows when a col is full Function Name Check_ifFree_test_2

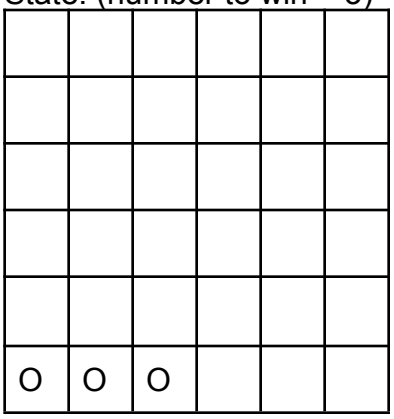
Input State: numToWin = 3 Row = 20 Column 20 	checkifFree returns true State of board does not change	Reason deals with a board on a bigger scale FunctionName:

		checkIfFree_test_with_doubleDigit_sizedBoard

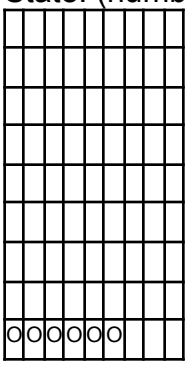
checkHorizWin:

Boolean checkHorizWin(BoardPosition pos, char p)

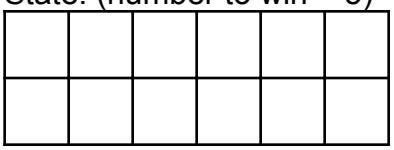
Input:	Output:	Reason: Test cases test

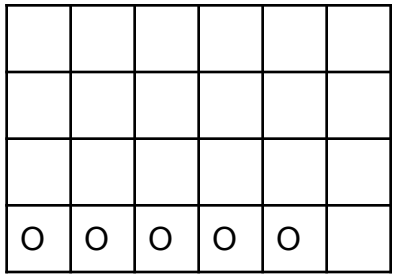
State: (number to win = 3)  P = 'o' pos.getRow = 0 pos.getCol = 2	checkHorizWin = true state of the board is unchanged	CheckHorizWin when token is last placed on the left side of the row Function Name: checkHorizWin_test_1

Boolean checkHorizWin(BoardPosition pos, char p)

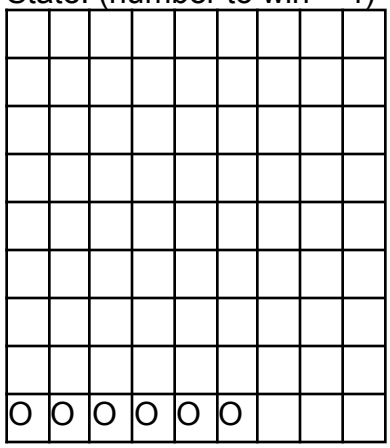
Input: State: (number to win = 5)  P = 'o' pos.getRow = 0 pos.getCol = 0	Output: checkHorizWin = true state of the board is unchanged	Reason: Test case test CheckHorizWin when token is last placed on the right side of the row Function Name: checkHorizWin_test_2

Boolean checkHorizWin(BoardPosition pos, char p)

Input: State: (number to win = 5) 	Output: checkHorizWin = true state of the board is unchanged	Reason: Test case test CheckHorizWin when token is last placed in the middle of the row. This test case is unique because the function must read right and left side of position.

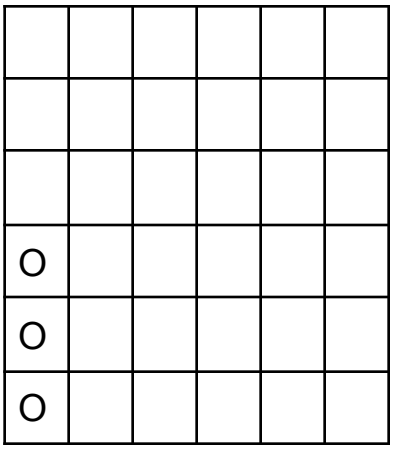
 P = ’o’ pos.getRow = 0 pos.getCol = 2		Function Name: checkHorizWin_test_3

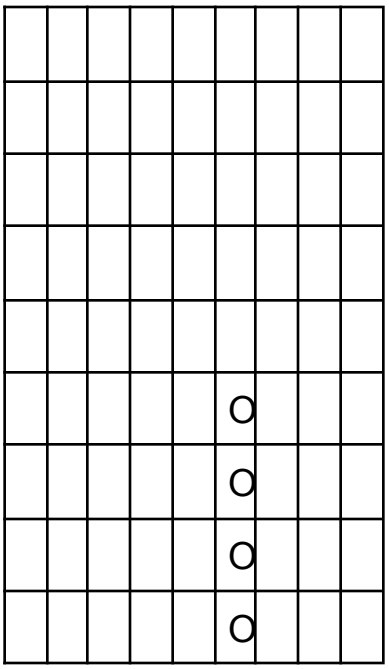
Boolean checkHorizWin(BoardPosition pos, char p)

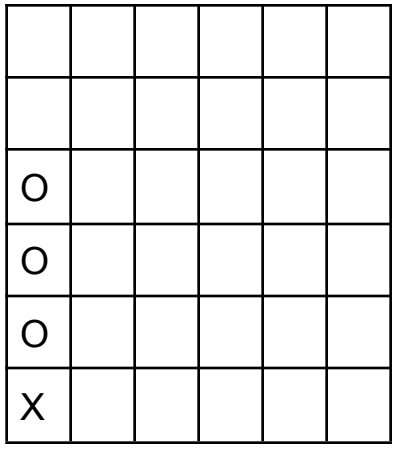
Input: State: (number to win = 4)  P = ’o’ pos.getRow = 0 pos.getCol = 3	Output: checkHorizWin = true state of the board is unchanged	Reason: Test case test CheckHorizWin when token is last placed in the middle of the row. This test case is unique because the function must read the right and left side of the position, it also adds up to a score greater than what is needed to win. Function Name: checkHorizWin_test_4

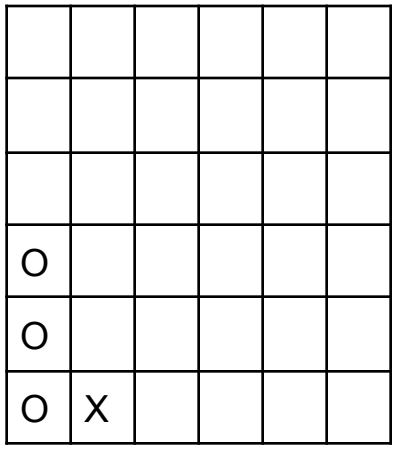
checkVertWin:

Input: Row = 6	Output CheckVertWin would return true	Reason: routine check vert win

Column =6 numsToWin =3  P = ‘O’	State of board is not changed	Function name checkvertWin_test_1

Input: Row = 9 Column = 9 NumstoWin = 4 	Output: CheckvertWin would return true State of board does not change	Reason: checks condition with a bigger win condition Function name checkvertWin_test_2

Input: State: numstoWin == 3 Row =6 Column = 6 	CheckVertWin should return True State of board does not change	Reason Test a vert with different characters to see if it could still read a win Name of function checkVertWin_test_3

Input: State: numsToWin 3 Row = 6 Column = 6  Last placed col=0 and row =2	CheckVertWin should return true State of board does not change	Reason: to ensure the method only checks one column of the last placed token Function name checkVertWin_test_4

checkDiagWin:

Boolean checkDiagWin(BoardPosition pos, char p)

Input:	Output:	Reason:Test checkDiagWin when token is last placed on the left bottom of the diag row. It is distinct because it tests from bottom left to top right.
State: (number to win = 3) |				| |		O	| |	O	O	| | O	O	O	| P = 'o' pos.getRow = 0 pos.getCol = 0	checkDiagWin = true state of the board is unchanged	Function Name: checkDiagWin_test_1

Boolean checkDiagWin(BoardPosition pos, char p)

Input:	Output:	Reason:Test checkDiagWin when token is last placed on the top right of the diag row. It is distinct because it tests from top right to bottom left.
State: (number to win = 3) |				| |		O	| |	O	O	| | O	O	O	| P = 'o' pos.getRow = 2 pos.getCol = 2	checkDiagWin = true state of the board is unchanged	Function Name: checkDiagWin_test_2

Boolean checkDiagWin(BoardPosition pos, char p)

Input:	Output:	Reason:Test checkDiagWin when token is last placed in the middle of the diag row. It is distinct because it has to test both sides from the middle, from bottom left to top right.
State: (number to win = 4) P = 'o' pos.getRow = 2 pos.getCol = 1	checkDiagWin = true state of the board is unchanged	Function Name: checkDiagWin_test_3

				O	
			O	O	
		O	O	O	
	O	O	O	O	
O	O	O	O	O	

Boolean checkDiagWin(BoardPosition pos, char p)

Input:	Output:	Reason:Test checkDiagWin when token is last placed on the top right of the diag row. It is distinct because it tests from top left to bottom right.
State: (number to win = 3) P = 'o' pos.getRow = 2 pos.getCol = 0	checkDiagWin = true state of the board is unchanged	Function Name: checkDiagWin_test_4

O			
O	O		
O	O	O	

Boolean checkDiagWin(BoardPosition pos, char p)

Input:	Output:	Reason:Test checkDiagWin when token is last placed on the bottom left of the diag row. It is distinct because it tests from top left to bottom right.
State: (number to win = 3) | | | | | |---|---|---|---| | O | | | | | O | O | | | | O | O | O | | P = 'o' pos.getRow = 0 pos.getCol = 2	checkDiagWin = true state of the board is unchanged	Function Name: checkDiagWin_test_5

Boolean checkDiagWin(BoardPosition pos, char p)

Input:	Output:	Reason:Test checkDiagWin when token is last placed in the middle of the diag row. It is distinct because it has to test both sides from the middle, from top left to bottom right.
State: (number to win = 4) | | | | | | | |---|---|---|---|---|---| | O | | | | | | | O | O | | | | | | O | O | O | | | | | O | O | O | O | | | | O | O | O | O | O | | P = 'o' pos.getRow = 1 pos.getCol = 2	checkDiagWin = true state of the board is unchanged	Function Name: checkDiagWin_test_6

Boolean checkDiagWin(BoardPosition pos, char p)

<table>
<tr>
<td>Input: numToWin:4
Row =6
Column = 6

<table><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>O</td><td></td><td></td><td></td><td></td><td>O</td></tr><tr><td>O</td><td>O</td><td></td><td></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></tr><tr><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></tr></table>Lastplacedtoken = row 1 col 2</td>
<td>CheckDiagWin returns true
State of the board is unchanged</td>
<td>Reason The test forms 5 diag lines and sees if it checks the only the last placed tokens possible win condition.

Function name:
checkDiagWin_test_7</td>
</tr>
</table>

checkTie:
Boolean checkTie(BoardPosition pos, char p)

<table>
<tr>
<td>Input: numtoWin =3
Row = 3
Column = 3

<table><tr><td>X</td><td>X</td><td>X</td></tr><tr><td>X</td><td>X</td><td>X</td></tr><tr><td>X</td><td>X</td><td>X</td></tr></table></td>
<td>checkTie would return true and the state of the board would not change</td>
<td>Reason Routine case of a filled board that would result in a Tie

Name of function

checkTie_test_1</td>
</tr>
</table>

Input: numToWin = 4 Row = 4	checkTie would return true and state of the	Reason check if a 4x4 board filled would still

<table>
<tr>
<td>Column = 4
<table><tr><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></tr><tr><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></tr></table></td>
<td>board would not change</td>
<td>yield a tie

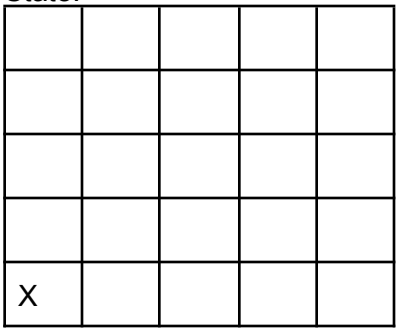
Name function:
checkTie_test_2</td>
</tr>
</table>

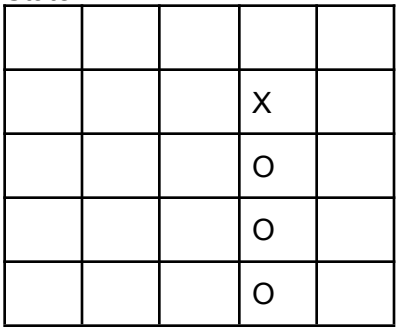
Input: Row = 20 Column = 20 numToWin = 20 Board is 20x20 filled with X's	Output checkTie would result in True	Reason: tests a case for a board of a bigger scale. Name of Function checkTie_test_3

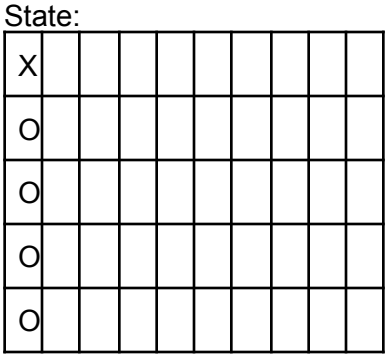
Input: Row = 100 Column = 100 numToWin = 25 Board is 100x100 filled with X's	Output checkTie would result in True	Reason: tests a case for a boundary Name of Function checkTie_test_4

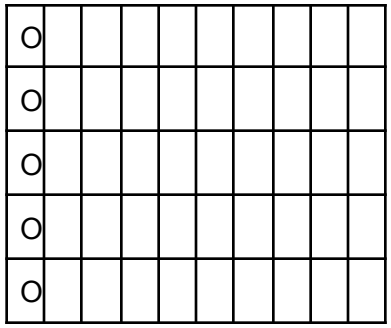
whatsAtPos:

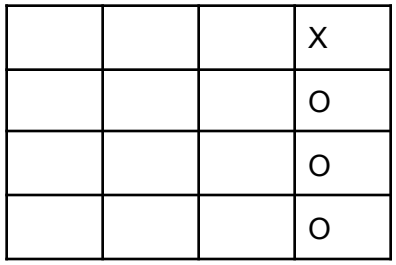
Char whatsAtPos(BoardPosition pos)

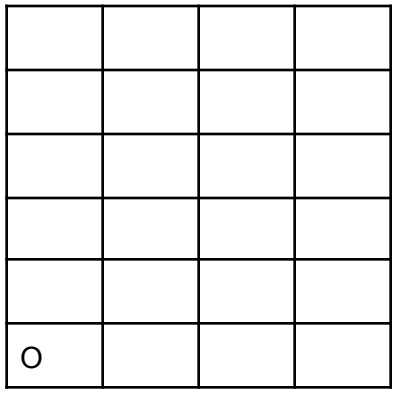
Input: State:  P = 'X' C = 0	Output: pos.getRow = 0 pos.getCol = 0 whatsAtPos(pos)= 'X' State of board is unchanged	Reason:This tests if the correct output is given for the coordinates 0,0 Function Name: whatsAtPos_Corner_test_1

Input: State:  P = 'X' C = 3	Output: pos.getRow = 3 pos.getCol = 3 whatsAtPos(pos)= 'X' State of board is unchanged	Reason:This tests if the correct output is given for the coordinates 3,3 Function Name: whatsAtPos_2_top_right_Corner_Pos

Input: State: 	Output: pos.getRow = 9 pos.getCol = 0 whatsAtPos(pos)= 'X' State of board is unchanged	Reason:This tests if the correct output is given for the coordinates 0,9 Function Name: whatsAtPos_3_top_left_Corner_Pos_biggerSize

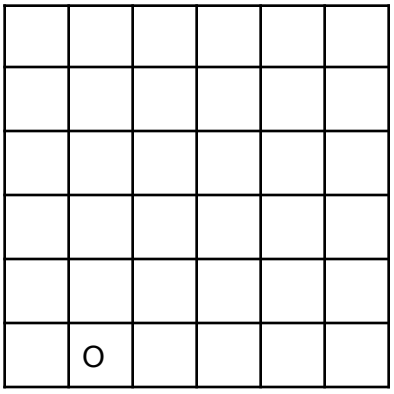
 P = ‘X’ C = 0		

Input: State:  C = 3	Output: pos.getRow = 2 pos.getCol = 3 whatsAtPos(pos)= ’O’ pos.getRow = 3 pos.getCol = 3 whatsAtPos(pos)= ’X’ State of board is unchanged	Reason:This tests if the correct output is given for the coordinates 3,2 and 3,3 Multiple calls tested Function Name: whatsAtPos_4_top_routine_ multipleCalls

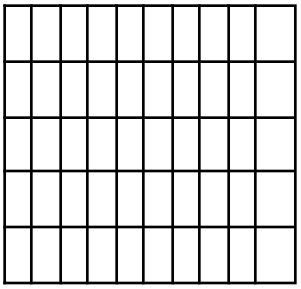
Input: State:  C = 2	Output: pos.getRow = 0 pos.getCol = 0 whatsAtPos(pos)= ’O’ State of board is unchanged	Reason:This tests if the correct output is given for the coordinates 0,0 Function Name: whatsAtPos_5_bottomLeft__ pos

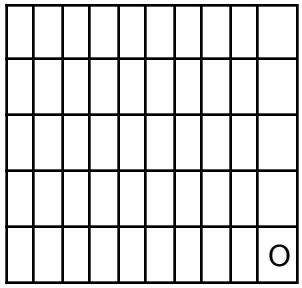
isPlayerAtPos:

boolean isPlayerAtPos(BoardPosition pos, char player)

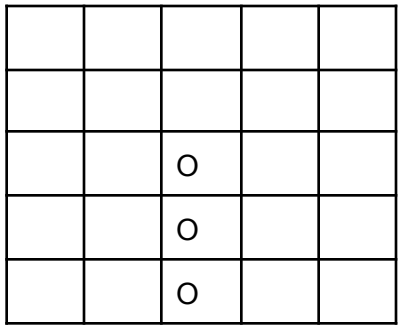
Input:	Output:	Reason: Test if token dropped in column 1 returns the correct character. Should return true.
 State: P = 'O' C = 1	pos.getRow = 0 pos.getCol = 1 isPlayerAtPos(pos,'O') = true State of board is unchanged	Function Name: isPlayerAtPos_test_1

boolean isPlayerAtPos(BoardPosition pos, char player)

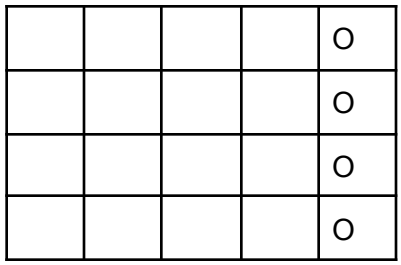
Input: numsToWin=3 Rows = 20 Cols = 20 "Board represent 20 x 20" State: 	Output: pos.getRow = 0 pos.getCol = 20 isPlayerAtPos(pos,'O') = true State of board is unchanged	Reason: Test if token dropped in column 20 returns the correct character. Should return true. It is distinct because it tests the boundaries of the board. Function Name: isPlayerAtPos_test_2_boundary_max

 P = 'O' C = 20		

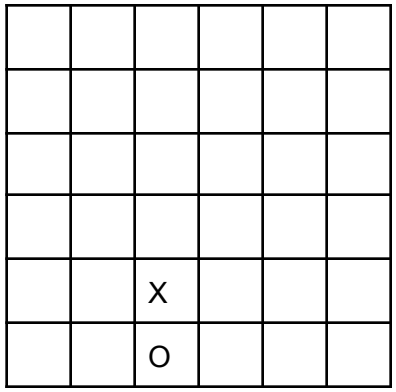
boolean isPlayerAtPos(BoardPosition pos, char player)

Input: numstoWin=3 rows=5 cols=5 State:  P = 'O' C = 2	Output: pos.getRow = 2 pos.getCol = 2 isPlayerAtPos(pos,'O') = true State of board is unchanged	Reason: Test middle of board. It is distinct because it tests placement in the middle of the board. Function Name: isPlayerAtPos_test_3_in_middle_of_board

boolean isPlayerAtPos(BoardPosition pos, char player)

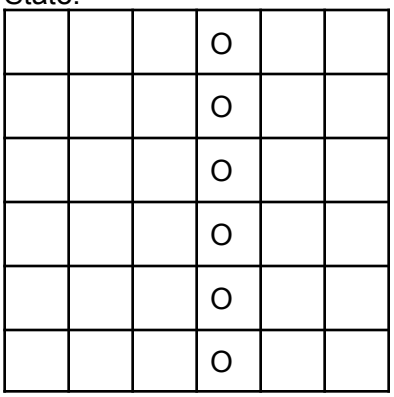
Input: numstoWin=3 rows=4 cols=4 State:  P = 'O' C = 3	Output: pos.getRow = 3 pos.getCol = 3 isPlayerAtPos(pos,'O') = true State of board is unchanged	Reason: Test top right of board. It is distinct because it is testing the top right of the board. Function Name: isPlayerAtPos_test_4_topRightofBoard

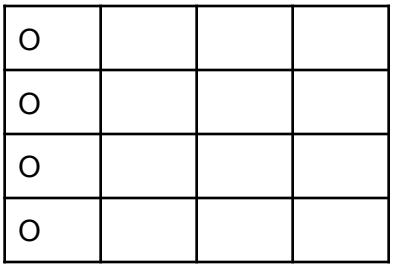
boolean isPlayerAtPos(BoardPosition pos, char player)

Input: numstoWin=3 Rows = 6 Cols = 6 State:  P = 'O'' C = 3	Output: pos.getRow = 3 pos.getCol = 3 isPlayerAtPos(pos,'O') = false State of board is unchanged	Reason: Test if character at pos 3,3 is character 'O'. Should return false, it is distinct because it tests a character besides the players. Function Name: isPlayerAtPos_test_5_differentCharAtSamePos

dropToken:

Void dropToken(char p, int c)

Input:	Output:	Reason:This tests if the correct output of whatsatPosition = O
State:  C = 3	whatsAtPos(pos)= 'O' State of board is unchanged	Function Name: dropToken_test_1

Input:	Output:	Reason:This tests if the correct output of whatsatPosition = O
State:  Lastpos = 0, 3	whatsAtPos(pos)= 'O' State of board is unchanged	Function Name: dropToken_test_firstCol_boundary_2

Input:		
State		