GameBoard(int nR, int nC, int nW)

<table>
<tr><th>Inputs</th><th>Expected Output</th><th>Reason for Test Case</th></tr>
<tr>
<td>State:
board = null
numRow = null
numCol = null
numToWin = null
nR = MIN_ROW
nC = MIN_COLUMNS
nW = MIN_TO_WIN</td>
<td>State:
board =
<table>
<tr><td></td><td>0</td><td>1</td><td>2</td></tr>
<tr><td>0</td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
</table>
numRow = 3
numCol = 3
numToWin = 3</td>
<td>This case represents the lower bound of a GameBoard’s size. In this case, numToWin, numCol, and numRow are all equal to their lowest possible values when the object is created.</td>
</tr>
</table>

GameBoard(int nR, int nC, int nW)

<table>
<tr><th>Inputs</th><th>Expected Output</th><th>Reason for Test Case</th></tr>
<tr>
<td>State:
board = null
numRow = null
numCol = null
numToWin = null
nR = MAX_ROWS
nC = MAX_COLUMNS
nW = 30</td>
<td>State:
board =
<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>…</td><td>99</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td>…</td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td>…</td><td></td></tr>
<tr><td>2</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></tr>
<tr><td>99</td><td></td><td></td><td></td><td>…</td><td></td></tr>
</table>
numRow = 100
numCol = 100
numToWin = 30</td>
<td>This case represents the upper bound of a GameBoard’s size. In this case, the number of rows and columns are set their maximum values but numToWin is set to a different, smaller value.</td>
</tr>
</table>

GameBoard(int nR, int nC, int nW)

Inputs	Expected Output	Reason for Test Case
State: board = null numRow = null numCol = null numToWin = null nR = 10 nC = 8 nW = 4	State: board = numRow = 10 numCol = 8 numToWin = 4	This case represents the general case for a GameBoard. It is a rectangular shape where the numToWin is not at either extreme, so all three inputs are different from each other. Most of the available layouts for a new board will be like this one.

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								
8								
9								

GameBoard(int nR, int nC, int nW)

Inputs	Expected Output	Reason for Test Case
State: board = null numRow = null numCol = null numToWin = null nR = MAX_ROWS nC = MIN_COLUMNS nW = MIN_TO_WIN	State: board = numRow = 100 numCol = 3 numToWin = 3	This case represents a wildly disproportionate GameBoard. numRow and numCol are set to their OPPOSITE extremes, and numToWin is set to its minimum. This tests the case where a board is as long or wide as possible.

	0	1	2
0			
1			
2			
…	…	…	…
99			

Boolean checkSpace(BoardPosition pos)

Input	Expected Outputs	Reason for Test Case
State: Pos.getRow() = -5 pos.getColumn() = 15	The state of the board is unchanged checkSpace = false	This tests the case where an invalid (out of range) position is passed in for evaluation.

	0	1	2	3	4	5	6	7
0								
1		X						
2			O					
3								
4					X			
5								
6								
7								
8								
9								

Boolean checkSpace(BoardPosition pos)

Input	Expected Outputs	Reason for Test Case
State: Pos.getRow = 2 Pos.getRow = 2	The state of the board is unchanged checkSpace = false	This tests the case where a position is checked that has already been played in. This is the case where a potentially valid input has been passed in, but it has already been used once before and will not be allowed again.

	0	1	2	3	4	5	6	7
0								
1		X						
2			O					
3								
4					X			
5								
6								
7								
8								
9								

Boolean checkSpace(BoardPosition pos)

Input	Expected Outputs	Reason for Test Case
State: Pos.getRow() = 6 Pos.getColumn() = 2	The state of the board is unchanged checkSpace = true	This tests the case where the method is called on an unoccupied position in range. This is the case where the position checked is valid.

	0	1	2	3	4	5	6	7
0								
1		X						
2			O					
3								
4					X			
5								
6								
7								
8								
9								

Boolean checkHorizontalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 1 lastPos.getColumn() = 1 player = X numToWin = 4	The state of the board is unchanged checkHorizontalWin = true	This tests the case where lastPos is less than numToWin positions away from the left side bound of the game board.

	0	1	2	3	4	5	6	7
0	X							O
1	X	X	X	X				O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkHorizontalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 3 lastPos.getColumn() = 5 player = O numToWin = 4	The state of the board is unchanged checkHorizontalWin = true	This tests the case where lastPos is less than numToWin positions away from the right side bound of the game board.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O	O	O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkHorizontalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 1 lastPos.getColumn() = 1 player = X numToWin = 4	The state of the board is unchanged checkHorizontalWin = true	This case tests for when a horizontal victory is achieved by aligning more than the required number of game pieces.

	0	1	2	3	4	5	6	7
0	X							O
1	X	X	X	X	X			O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkHorizontalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 6 lastPos.getColumn() = 5 player = O numToWin = 4	The state of the board is unchanged checkHorizontalWin = false	This tests the case where a player has placed their marker in a line of their own markers, but there are not enough markers aligned to achieve victory.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O	O	O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkVerticalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 2 lastPos.getColumn() = 7 player = O numToWin = 4	The state of the board is unchanged checkVerticalWin = true	This tests the case where lastPos is less than numToWin positions away from the upper bound of the game board.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		O
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkVerticalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 9 lastPos.getColumn() = 6 player = O numToWin = 4	The state of the board is unchanged checkVerticalWin = true	This tests the case where lastPos is less than numToWin positions away from the lower bound of the game board.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O				O	

Boolean checkVerticalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 4 lastPos.getColumn() = 6 player = O numToWin = 4	The state of the board is unchanged checkVerticalWin = false	This tests the case where the player has aligned some of their markers vertically but identifies that they have failed to align enough to win.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O		O	O
4							O	
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkVerticalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 5 lastPos.getColumn() = 6 player = O numToWin = 4	The state of the board is unchanged checkVerticalWin = true	This tests the case where the player has won by aligned more than the required number of game pieces in a vertical line.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O		O	O
4							O	
5							O	
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 1 lastPos.getColumn() = 6 player = O numToWin = 4	The state of the board is unchanged checkDiagonalWin = true	Tests the case where the user wins along the minor diagonal, but last pos is at least numToWin game pieces from the top right corner of the board.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X			O	O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 9 lastPos.getColumn() = 7 player = O numToWin = 4	The state of the board is unchanged checkDiagonalWin = true	Tests the case where the user wins along the minor diagonal, but last pos is at least numToWin game pieces from the bottom right corner of the board.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					O

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 1 lastPos.getColumn() = 1 player = X numToWin = 4	The state of the board is unchanged checkDiagonalWin = true	Tests the case where the user wins along the minor diagonal, but last pos is at least numToWin game pieces from the top left corner of the board.

	0	1	2	3	4	5	6	7
0	X							O
1	X	X	X	X				O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 8 lastPos.getColumn() = 1 player = X numToWin = 4	The state of the board is unchanged checkDiagonalWin = true	Tests the case where the user wins along the minor diagonal, but last pos is at least numToWin game pieces from the bottom left corner of the board.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8		X					O	
9	X		O					

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 8 lastPos.getColumn() = 1 player = O numToWin = 4	The state of the board is unchanged checkDiagonalWin = false	This tests the case where the lastPosition is in a line of characters that borders the bound of the game board at two points.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O		O	O
4								
5								
6				X	O		O	
7	O		X			O	O	
8		O					O	
9	X		O					

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 4 lastPos.getColumn() = 4 player = X numToWin = 4	The state of the board is unchanged checkDiagonalWin = false	Tests the case where the lastPos is in a line of characters along the main diagonal, but there are not enough markers in line to achieve a victory.

	0	1	2	3	4	5	6	7
0	X							O
1	X		X	X				O
2			X			O		
3				X	O		O	O
4					X			
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input	Expected Outputs	Reason for Test Case
State: lastPos.getRow() = 1 lastPos.getColumn() = 1 player = X numToWin = 4	The state of the board is unchanged checkDiagonalWin = true	Tests the case where the player wins by aligning more than the required number of markers along the main diagonal

	0	1	2	3	4	5	6	7
0	X							O
1	X	X	X	X				O
2			X			O		
3				X	O		O	O
4					X			
5								
6				X	O		O	
7	O		X			O	O	
8							O	
9	X		O					

Boolean checkDraw(void)

Input	Expected Outputs	Reason for Test Case
State: | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | A | B | C | D | E | F | G | H | | 1 | I | J | A | B | C | D | E | F | | 2 | G | H | I | J | A | B | C | D | | 3 | E | F | G | H | I | J | A | B | | 4 | C | D | E | F | G | H | I | J | | 5 | A | B | C | D | E | F | G | H | | 6 | I | J | A | B | C | D | E | F | | 7 | G | H | I | J | A | B | C | D | | 8 | E | F | G | H | I | J | A | B | | 9 | C | D | E | F | G | H | I | J |	The state of the board is unchanged checkDraw = true	This case represents the situation where a draw has resulted. There is no winner

Boolean checkDraw(void)

Input	Expected Outputs	Reason for Test Case
State: | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | X | | | | | | | | | 1 | | | | | O | | | | | 2 | | | | | | | | | | 3 | | | | | O | | | | | 4 | | | X | | O | | | | | 5 | | | | X | | | | | | 6 | | | | | | | | | | 7 | | | | | | | | | | 8 | | | | | | | | | | 9 | | | | | | | | |	The state of the board is unchanged checkDraw = false	This case represents a case where no draw is present and some players have already made moves on the board.

Boolean checkDraw(void)

Input	Expected Outputs	Reason for Test Case
State: | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | | | | | | | | | | 1 | | | | | | | | | | 2 | | | | | | | | | | 3 | | | | | | | | | | 4 | | | | | | | | | | 5 | | | | | | | | | | 6 | | | | | | | | | | 7 | | | | | | | | | | 8 | | | | | | | | | | 9 | | | | | | | | |	The state of the board is unchanged checkDraw = false	This case represents a situation when the method is called on an empty board where no players have made a move.

Boolean checkDraw(void)

Input	Expected Outputs	Reason for Test Case
State: | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | X | A | B | C | D | E | F | G | | 1 | H | X | I | A | B | C | D | E | | 2 | F | G | X | H | I | A | B | C | | 3 | D | E | F | X | G | H | I | A | | 4 | B | C | D | E | X | F | G | H | | 5 | I | A | B | C | D | X | E | F | | 6 | G | H | I | A | B | C | X | D | | 7 | E | F | G | H | I | A | B | X | | 8 | C | D | E | F | G | H | I | A | | 9 | B | C | D | E | F | G | H | I | numToWin = 4	The state of the board is unchanged checkDraw = true	This case represents a situation where there is a draw, but a win condition is present on the board. This is to illustrate that the method should return true when the board is full, even if the game has not truly ended in a draw.

char whatsAtPos(BoardPosition pos)

Input	Expected Outputs	Reason for Test Case
State: Pos.getRow = 5 Pos.getColumn = 15	The state of the board is unchanged whatsAtPos = ‘ ‘	This tests the case where a position with a valid row is passed in, but its column over the allowed maximum.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

char whatsAtPos(BoardPosition pos)

Input	Expected Outputs	Reason for Test Case
State: Pos.getRow = 0 Pos.getColumn = 0	The state of the board is unchanged whatsAtPos = ‘ ‘	This tests the case where the position is empty and in range.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

char whatsAtPos(BoardPosition pos)

Input	Expected Outputs	Reason for Test Case
State: | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | | | | | | | | | | 1 | | | | | | | | | | 2 | | | X | | | | O | | | 3 | | | | | | | | | | 4 | | | | X | O | | | | | 5 | | | | | | O | | | | 6 | | | X | | | | | | | 7 | | | | | | | | | | 8 | | | | | | | | | | 9 | | | | | | | | | Pos.getRow = -3 Pos.getRow = -3	The state of the board is unchanged whatsAtPos = ‘ ’	This case represents a situation when a position that has negative coordinates is passed in.

char whatsAtPos(BoardPosition pos)

Input	Expected Outputs	Reason for Test Case
State: | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | | | | | | | | | | 1 | | | | | | | | | | 2 | | | X | | | | O | | | 3 | | | | | | | | | | 4 | | | | X | O | | | | | 5 | | | | | | O | | | | 6 | | | X | | | | | | | 7 | | | | | | | | | | 8 | | | | | | | | | | 9 | | | | | | | | | Pos.getRow = 15 Pos.getColumn = 5	The state of the board is unchanged whatsAtPos = ‘ ‘	Tests the case where a position with a column over the allowed maximum is passed in.

char whatsAtPos(BoardPosition pos)

Input	Expected Outputs	Reason for Test Case
State: Pos.getRow = 2 Pos.getColumn = 2	The state of the board is unchanged whatsAtPos = 'X'	This case represents a situation where a position is looked at that has a player marker placed in it.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

Boolean isPlayerAtPos(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = 'X' Pos.getRow = 0 Pos.getColumn = 0	The state of the board is unchanged isPlayerAtPos = false	The case represents a situation when the method is called and to check for a player that does exist on the GameBoard but in a position that does not contain that player. This checks to make sure that the method is not returning a positive value just because a piece is present on the board somewhere.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

Boolean isPlayerAtPos(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = 'X' Pos.getRow=2 Pos.getColumn=2	The state of the board is unchanged isPlayerAtPos = true	This tests the situation where a position is passed in with a non-blank character that matches the data in the GameBoard.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

Boolean isPlayerAtPos(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = 'X' Pos.getRow = 2 Pos.getColumn = 6	The state of the board is unchanged isPlayerAtPos = false	This represents a situation where the method is called on a taken position and given a player character as input, but the input that is given does not match the character that is occupying the space.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

Boolean isPlayerAtPos(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = ' ' pos.getRow = 2 pos.getColumn = 2	The state of the board is unchanged isPlayerAtPos = false	Tests the case where an occupied space is checked to see if it is empty of not.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

Boolean isPlayerAtPos(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = ' ' Pos.getRow = 0 Pos.getColumn = 0	The state of the board is unchanged isPlayerAtPos = true	This checks the situation where an empty position is checked against a character to represent an empty space.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

void placeMarker(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = 'X' Pos.getRow = 0 Pos.getColumn = getNumColumns	State:	This tests the case where the marker is added at the first quadrant's bound.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

	0	1	2	3	4	5	6	7
0								X
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

void placeMarker(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = 'X' Pos.getRow = getNumRows() Pos.getColumn = getNumColumns()	State:	This tests the case where the marker is added at the fourth quadrant's bound.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								X

void placeMarker(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = 'X' Pos.getRow = 0 Pos.getColumn = 0	State:	This tests the case where the marker is added at the second quadrant's bound.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

	0	1	2	3	4	5	6	7
0	X							
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

void placeMarker(BoardPosition pos, char player)

Input	Expected Outputs	Reason for Test Case
State: Player = 'X' Pos.getRow = getNumRows Pos.getColumn = 0	State:	This tests the case where a player is added at the third quadrant's bound.

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9								

	0	1	2	3	4	5	6	7
0								
1								
2			X				O	
3								
4				X	O			
5						O		
6			X					
7								
8								
9	X							

void placeMarker(BoardPosition pos, char player)

<table>
<tr><th>Input</th><th>Expected Outputs</th><th>Reason for Test Case</th></tr>
<tr>
<td>State:
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td>X</td><td></td><td></td><td></td><td>O</td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td>X</td><td>O</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td>O</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
Player = ‘Z’
Pos.getRow = 2
Pos.getColumn = 4
</td>
<td>State:
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td>X</td><td></td><td>Z</td><td></td><td>O</td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td>X</td><td>O</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td>O</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td>
<td>The case tests the situation where a new player is added to the game board.</td>
</tr>
</table>