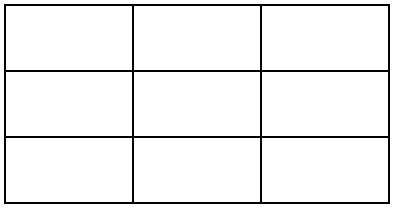
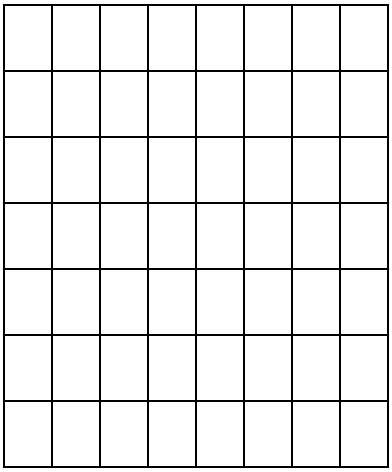
Test Cases

GameBoard(int rows, int cols, int winAmt)

Input: rows = 3 cols = 3 winAmt = 3	Output: 	Reason: Unique because testing minimum bounds Function Name: constructor_3_3_3

Input: Rows = 100 Cols = 100 WinAmt = 25	Output: (Board of size 100x100 created)	Reason: Unique because testing maximum bounds Function Name: constructor_100_100_25

Input: Rows = 7 Cols = 8 WinAmt = 5	Output: 	Reason: Unique because testing when bounds are not the same for rows and columns Function Name: constructor_7_8_5

boolean checkSpace(BoardPosition pos)

Input: State: | | 0 | 1 | 2 | | --- | --- | --- | --- | | 0 | | | | | 1 | | | | | 2 | | | | pos.getRow = 0 pos.getCol = 0	Output: checkSpace = true State of board unchanged	Reason: Unique because testing empty board, empty space Function Name: checkSpace_emptyboard

Input: State: | | 0 | 1 | 2 | | --- | --- | --- | --- | | 0 | X | | | | 1 | | | | | 2 | | | | pos.getRow = 0 pos.getCol = 0	Output: checkSpace = false State of board unchanged	Reason: Unique because position is filled already Function Name: checkSpace_notavailable

Input: State: | | 0 | 1 | 2 | | --- | --- | --- | --- | | 0 | | | | | 1 | | | | | 2 | | | | pos.getRow = 3 pos.getCol = 3	Output: checkSpace = false State of board unchanged	Reason: Unique because we are testing the out of bounds (not a precondition) Function Name: checkSpace_outOfBounds

boolean checkHorizontalWin(BoardPosition lastPos, char player)

Input:	Output:	Reason:
numToWin = 4 | | 0 | 1 | 2 | 3 | | --- | --- | --- | --- | --- | | 0 | X | X | X | | | 1 | | | | | | 2 | | | | | pos.getRow = 0 pos.getCol = 2 P = x	checkHorizontalWin = false State of board unchanged	Unique because we are checking to make sure the number in a row is equal to numToWin and not a hardcoded number such as 3 Function Name: checkHorizontalWin_noWin

Input:	Output:	Reason:
numToWin = 4 | | 0 | 1 | 2 | 3 | | --- | --- | --- | --- | --- | | 0 | X | X | X | X | | 1 | | | | | | 2 | | | | | pos.getRow = 0 pos.getCol = 3 P = x	checkHorizontalWin = true State of board unchanged	Unique because it checks a win made by a character on the right (has to count to the left) Function Name: checkHorizontalWin_WinEnd

Input:	Output:	Reason:
numToWin = 4 | | 0 | 1 | 2 | 3 | | --- | --- | --- | --- | --- | | 0 | X | X | X | A | | 1 | | | | | | 2 | | | | | pos.getRow = 0 pos.getCol = 3 P = A	checkHorizontalWin = false State of board unchanged	Unique because it checks to see if other characters beside the placed character don't cause a win Function Name: checkHorizontalWin_noWinEnd_difPlayer

Input: numToWin = 4 |				
	0	1	2	3
0	X	X	X	X
1				
2				| pos.getRow = 0 pos.getCol = 0 P = X	Output: checkHorizontalWin = true State of board unchanged	Reason: Unique because it checks a win made a character on the left (has to count to the right) Function Name: checkHorizontalWin_WinStart
---	---	---

boolean checkVerticalWin(BoardPosition lastPos, char player)

Input: numToWin = 4 			
	0	1	2
0	X		
1	X		
2	X		
3			| pos.getRow = 2 pos.getCol = 0 P = x	Output: checkVerticalWin = false State of board unchanged	Reason: Unique because it checks to make sure the number in a row is equal to numToWin and not a hardcoded value such as 3 Function Name: checkVerticalWin_noWin
---	---	---

Input: numToWin = 4 			
	0	1	2
0	X		
1	X		
2	X		
3	X		| pos.getRow = 3	Output: checkVerticalWin = true State of board unchanged	Reason: Unique because it checks a win made a character on top of it (has to count up) Function Name: checkVerticalWin_WinEnd
---	---	---

pos.getCol = 0 P = x		

Input: numToWin = 4 pos.getRow = 3 pos.getCol = 0 P = A	Output: checkVerticalWin = false State of board unchanged	Reason: Unique because it checks to see if other characters above it don't cause a win Function Name: checkVerticalWin_WinEnd_difPlayer

	0	1	2
0	X		
1	X		
2	X		
3	A		

Input: numToWin = 4 pos.getRow = 0 pos.getCol = 0 P = X	Output: checkVerticalWin = true State of board unchanged	Reason: Unique because it checks a win made by a character on top (has to count down) Function Name: checkVerticalWin_WinStart

	0	1	2
0	X		
1	X		
2	X		
3	X		

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input:	Output:	Reason:
numToWin = 4 | | 0 | 1 | 2 | | 0 | X | | | | 1 | | X | | | 2 | | | X | | 3 | | | | pos.getRow = 2 pos.getCol = 2 P = x	checkDiagonalWin = false State of board unchanged	Unique because it checks to make sure the number in a row is equal to numToWin and not a hardcoded value such as 3 Function Name: checkDiagonalWin_backSlash_noWin

Input:	Output:	Reason:
numToWin = 4 | | 0 | 1 | 2 | 3 | | 0 | X | | | | | 1 | | X | | | | 2 | | | X | | | 3 | | | | X | pos.getRow = 2 pos.getCol = 2 P = x	checkDiagonalWin = true State of board unchanged	Unique because it checks a win made by a character in the middle and a diagonal going like '\' (has to count to the top left and bottom right) Function Name: checkDiagonalWin_Win_BackSlash_Center

Input:
numToWin = 4

	0	1	2	3
0	X			
1		X		
2			X	
3				A

pos.getRow = 3
pos.getCol = 3
P = A

Output:
checkDiagonaWin = false

State of board unchanged

Reason:
Unique because it checks to see if other characters to the top left don't cause a win

Function Name:
checkDiagonalWin_Win_BackSlash_difPlayer

Input:
numToWin = 4

	0	1	2	3
0				X
1			X	
2		X		
3	X			

pos.getRow = 2
pos.getCol = 1
P = X

Output:
checkDiagonalWin = true

State of board unchanged

Reason:
Unique because it checks a win made by a character in the middle and going like '/' (has to count to the bottom left and top right)

Function Name:
checkDiagonalWin_win_forwardSlash_difPlayer

Input:
numToWin = 4

	0	1	2	3
0				A
1			X	
2		X		
3	X			

Output:
checkDiagonaWin = false

State of board unchanged

Reason:
Unique because it checks to see if other characters don't cause a win for the other diagonal

Function Name:
checkDiagonalWin_Win_ForwardSlash_difPlayer

pos.getRow = 0 pos.getCol = 3 P = A		

Input: numToWin = 4 | | 0 | 1 | 2 | | --- | --- | --- | --- | | 0 | | | X | | 1 | | X | | | 2 | X | | | | 3 | | | | pos.getRow = 0 pos.getCol = 2 P = x	Output: checkDiagonalWin = false State of board unchanged	Reason: Unique because it checks to make sure the number in a row is equal to numToWin for other diagonal Function Name: checkDiagonalWin_forwardSlash_noWin

Input: numToWin = 4 | | 0 | 1 | 2 | 3 | 4 | | --- | --- | --- | --- | --- | --- | | 0 | | | | | X | | 1 | | | | | | | 2 | | | X | | | | 3 | | X | | | | | 4 | X | | | | | pos.getRow = 0 pos.getCol = 2 P = x	Output: checkDiagonalWin = false State of board unchanged	Reason: Unique because it checks to make sure it doesn't check the whole diagonal and the players must be next to each other (loop stops when a blank space is reached) Function Name: checkDiagonalWin_forwardSlash_noWin_gap

boolean checkForDraw()

Input:	Output:	Reason:
<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>	checkForDraw = false	Unique because it is checking when board is empty, draw should be false Function Name: checkForDraw_emptyBoard

Input:	Output:	Reason:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>X</td><td>O</td><td>X</td></tr><tr><td>1</td><td>X</td><td>X</td><td>O</td></tr><tr><td>2</td><td>O</td><td>X</td><td>O</td></tr></table>	checkForDraw = true	Unique because we are checking when board is full and no win has occurred Function Name: checkForDraw_fullBoard_noWin

Input:	Output:	Reason:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>X</td><td>X</td><td></td></tr><tr><td>1</td><td>X</td><td>O</td><td></td></tr><tr><td>2</td><td>O</td><td>X</td><td></td></tr></table>	checkForDraw = false	Unique because we are checking when board it has characters in every position except the last column (makes sure it checks every column) and can stop a 1 off error. Function Name: checkForDraw_checkCol

Input:	Output:	Reason:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>X</td><td>X</td><td>O</td></tr><tr><td>1</td><td>X</td><td>O</td><td>X</td></tr><tr><td>2</td><td></td><td></td><td></td></tr></table>	checkForDraw = false	Unique because we are checking when board it has characters in every position except the last row (makes sure it checks every column) and can stop a 1 off error. Function Name: checkForDraw_checkRow

char whatsAtPos(BoardPosition pos)

Input:	Output:	Reason:
<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> Pos.getRow = 0 Pos.getCol = 0	whatsAtPos = ‘ ‘	Unique because we are checking when nobody is at a space (space is not a character as it is not defined in our map) Function Name: whatsAtPos_emptySpace

Input:	Output:	Reason:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>X</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> Pos.getRow = 0 Pos.getCol = 0	whatsAtPos = ‘X‘	Unique because we are checking when a player is at the first position (testing for one off errors) Function Name: whatsAtPos_player_startOfBoard

Input:	Output:	Reason:
<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>X</td></tr></table> Pos.getRow = 2 Pos.getCol = 2	whatsAtPos = ‘X‘	Unique because we are checking when a player is at the last position on the board (testing for one off errors) Function Name: whatsAtPos_player_endOfBoard

Input:

	0	1	2
0	Z	A	B
1	V	L	C
2	P	O	X

Pos.getRow = 1
Pos.getCol = 1

Output:

whatsAtPos = ‘L‘

Reason:
Unique because we are checking when a player is surrounded by other different characters (base case where we test for something in the middle)

Function Name:
whatsAtPos_player_fullBoard

Input:

	0	1	2
0	Z	A	B
1	V		C
2	P	O	X

Pos.getRow = 1
Pos.getCol = 1

Output:

whatsAtPos = ‘ ‘

Reason:
Unique because we are checking when an empty spot is surrounded by other different characters (our space is not a character in our memory efficient implementation)

Function Name:
whatsAtPos_noPlayer_fullBoard

boolean isPlayerAtPos(BoardPosition pos, char player)

Input:

	0	1	2
0			
1			
2			

Pos.getRow = 0
Pos.getCol = 0
P = X

Output:
isPlayerAtPos = false

Reason:
Unique because we are testing the Base Case where there is no player at pos

Function Name:
isPlayerAtPos_emptyBoard

<table>
<tr>
<td>Input:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>X</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>Pos.getRow = 0
Pos.getCol = 0
P = X</td>
<td>Output:
isPlayerAtPos = true</td>
<td>Reason:
Unique because we are testing the case where character is in the right position at the start of the board

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

<table>
<tr>
<td>Input:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>A</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>Pos.getRow = 0
Pos.getCol = 0
P = X</td>
<td>Output:
isPlayerAtPos = false</td>
<td>Reason:
Unique because we are testing the case where the wrong character is in the position, checks to make sure the player at the position is the right one.

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

<table>
<tr>
<td>Input:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>A</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>C</td></tr></table>Pos.getRow = 2
Pos.getCol = 2
P = C</td>
<td>Output:
isPlayerAtPos = true</td>
<td>Reason:
Unique because we are testing the last position in the board (bounds testing)

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

<table>
<tr>
<td>Input:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>A</td><td>C</td><td>X</td></tr><tr><td>1</td><td>A</td><td>O</td><td>I</td></tr><tr><td>2</td><td>C</td><td>M</td><td>M</td></tr></table>Pos.getRow = 0
Pos.getCol = 1
P = C</td>
<td>Output:
isPlayerAtPos = true</td>
<td>Reason:
Unique because we are testing the board when it is full of characters

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

void placeMarker(BoardPosition marker, char player)

<table>
<tr>
<td>Input:
P = X
Pos.getRow = 0
Pos.getCol = 0
<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></td>
<td>Output:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>X</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></td>
<td>Reason: Unique because we are testing base case where board is empty and we are placing in the first position of board

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

<table>
<tr>
<td>Input:
P = X
Pos.getRow = 0
Pos.getCol = 1
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>X</td><td></td><td>X</td></tr><tr><td>1</td><td>X</td><td>O</td><td>O</td></tr><tr><td>2</td><td>O</td><td>X</td><td>O</td></tr></table></td>
<td>Output:
<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>X</td><td>X</td><td>X</td></tr><tr><td>1</td><td>X</td><td>O</td><td>O</td></tr><tr><td>2</td><td>O</td><td>X</td><td>O</td></tr></table></td>
<td>Reason: Unique because we are testing when the board is one off from being full

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

Input:
P = A
Pos.getRow = 0
Pos.getCol = 1

	0	1	2
0	X		X
1	X	O	O
2	O	X	O

Output:

	0	1	2
0	X	A	X
1	X	O	O
2	O	X	O

Reason: Unique because we are testing a character besides X or O which could be hardcoded from other projects

Function Name: placeMarker_fullBoard_newCharacter

Input:
P = X
Pos.getRow = 2
Pos.getCol = 2

	0	1	2
0			
1			
2			

Output:

	0	1	2
0			
1			
2			X

Reason: Unique because we are testing base case where board is empty and we are placing at end of board

Function Name: placeMarker_emptyBoard_end

Input:
P = X
Pos.getRow = 1
Pos.getCol = 1

	0	1	2
0			
1			
2			X

Output:

	0	1	2
0			
1		X	
2			X

Reason: Unique because we are testing when a character is used for the second time.

Function Name: placeMarker_secondUsage