Input: State: (no board has been created entRow = 3 entColumn = 3 amtToWin = 3	State: (board of size was created) enteredRow = 3 enteredColumn = 3 enteredATW = 3 0 1 2 int amtToWin)						Reason: This test was made to assess the ability of the constructor to make a board of various single digit size. Function Name: construct_Small_Board_Test		
Input: State: (no board has been created) entRow = 8 entColumn = 8 amtToWin = 5	Outp Stat ente		ard of w = 8	size \ ; = 8	was o	5	d)	7	Reason: This test was made to assess the ability of the constructor to make a board of equal size. Function Name: construct_Square_Board_Test

GameBoard(int entRow, int entColumn, int amtToWin)

GameBoard(int entRow, int entColumn, int amtToWin)																					
Input:		utput: Reason:								Reason: This test was made to											
State: (no board has been created)	ent ent	State: (board of size was created) enteredRow = 100 enteredColumn = 100 enteredATW = 11											assess the ability of the constructor to make a board the test the max possible size.								
entRow = 100 entColumn = 100 amntToWin = 11		(20 by 20 table times 5, can not display a table of size 100)												Function Name: construct_Large_Board_T							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	est

boolean c	necklfFre	e(int col)			
Input: 'col' = 2 State: 0	1	2 O X O X	3	Output: false State: Unchanged	Reason: This function tests that column is not free when it is completely full. Function Name: checkIfFree_Full_Test
boolean c	hecklfFre	e(int col)			
Input: 'col' = 1 State: 0	1	2	3	Output: true State: Unchanged	Reason: This function tests that it accurately assess that there is a free spot in a column when it has not been populated. Function Name: checkIfFree_Empty_Test
boolean c	necklfFre	e(int col)			
Input: 'col' =1				Output: true	Reason: This function tests that it accurately assess that there is a free spot in a
State: 0	1 X	2	3	State: Unchanged	column when it has been populated but not fully. Function Name: checkIfFree_Partial_Test

boolean ch	neckHoriz [*]	Win(Boar	dPosition p	oos, char token)			
Input: pos.getRi pos.getCi 'token' =	olumn = 3	3		Output: True State: Unchanged	Reason: This function tests when the winning piece is placed on the lowest row to make sure it results correctly.		
State: 0	1	2	3		Function Name: checkHorizWin_Low_Test		
0	0	0					
X	Х	Х	Х				
boolean ch	neckHoriz	Win(Boar	dPosition բ	pos, char token)			
Input: pos.getRi pos.getCi 'token' =	olumn = 0)		Output: True State: Unchanged	Reason: This function tests when the winning piece is placed on the highest row to make sure it results correctly.		
State:	1	2	3		Function Name: checkHorizWin_High_Test		
Х	Х	Х	Х				
0	0	0	Х				
X	0	Х	0				
0	0	0	Х				
boolean ch	neckHoriz	Win(Boar	dPosition p	oos, char token)			
Input: pos.getRe	ow = 1 olumn = 2			Output: True State: Unchanged	Reason: This function tests when the winning piece is placed in the middle of other like pieces to ensure that it checks		
State:		•	•		both directions for a win.		
0	1	2	3		Function Name: checkHorizWin_Middle_Test		
	0		0				
Х	Х	Х	Х				
0	0	Х	0				
				•	1		

Input: pos.getRow = 2 pos.getColumn = 2 'token' = 0 State: 0 1 2	3	Output: false State: Unchanged	Reason: This function tests when a horizontal win condition is not fulfilled Function Name: checkHorizWin_False_Test		
x o o					
x x x	0				
0 0 0	Х				
Input: pos.getRow = 3 pos.getColumn = 0 'token' = X State:	dPosition po	Output: True State: Unchanged	Reason: This function tests when the winning piece is placed on the top of the furthest left column to make sure it results correctly.		
0 1 2 X	3		Function Name: checkVertWin_Left_Test		
X O	+				
хо					
x o	+ -				
Input: pos.getRow = 3 pos.getColumn = 3 'token' = X State: 0 1 2 O O	3 X X X X	Output: True State: Unchanged	Reason: This function tests when the winning piece is placed on the top of the furthest right column to make sure it results correctly. Function Name: checkVertWin_Right_Test		

boolean checkHorizWin(BoardPosition pos, char token)

Input: pos.getF pos.getC 'token' =	Row = 2 Column =		·	Output: True State: Unchanged	Reason: This function tests when the winning piece is not placed on the top row to make sure it results correctly.			
State: 0	1	2	3		Function Name: checkVertWin_Mid_Test			
		Х						
	0	Х						
	0	Х						
ooolean o	checkVert	:Win(Boa	rdPosition p	os, char token) Output:	Reason:			
pos.getF pos.getC 'token' =	Column =	2		False State: Unchanged	This function tests when the winning number of pieces are placed in a column but are different tokens therefore not winning to make sure it			
State: 0	1	2	3		results correctly. Function Name: checkVertWin_False_Test			
	Х	0			Greekvertviii_i alse_iest			
	0	Х						
	0	Х						
noolean c	·heckDia	nWin(Bos	ardPosition r	pos, char token)				
Input: pos.getF	Row = 0 Column =		arai osition į	Output: True State: Unchanged	Reason: This function tests when the winning piece is placed on at the bottom left end of a diagonal.			
State:	1	2	3		Function Name: checkDiagWin_BottomL_Test			
	Х	0						
	0	X	0					
<u> </u>	-	-						

boolean checkVertWin(BoardPosition pos, char token)

boolean o	<u>checkDia</u>	gWin(Boa	ardPosition p	os, char token)	
Input: pos.geti pos.get 'token' =	Row = 3 Column =			Output: True State: Unchanged	Reason: This function tests when the winning piece is placed on at the top right end of a diagonal.
State:	1	2	3		Function Name: checkDiagWin_TopR_Test
	<u> </u>		T _o		CheckDiagwin_TopK_Test
	X	0	X		
	0	X	X		
	0	X	0		
boolean	checkDia	aWin(Boa	ardPosition r	os, char token)	-
Input: pos.geti	Row = 3 Column =		2.0.00.0.0.	Output: True State: Unchanged	Reason: This function tests when the winning piece is placed on at the top left end of a diagonal.
State: 0	1	2	3		Function Name: checkDiagWin_TopL_Test
X					
0	Х				
X	0	Х			
0	0	Х			
boolean o	checkDia _s	gWin(Boa	ardPosition p	os, char token)	
Input: pos.geti pos.geti 'token' =	Column =	3		Output: True State: Unchanged	Reason: This function tests when the winning piece is placed on at the bottom right end of a diagonal.
State:		•			Function Name:
	1	2	3		checkDiagWin_BottomR_Test
	X	0			
	0	X			
	0	Х	Х		
			I		

Input: pos.getR pos.getC 'token' =	olumn = 1			Output: True State: Unchanged	Reason: This function tests when the winning piece is placed in the middle of a right diagonal to make sure it results correctly.
State: 0	1	2	3		Function Name: checkDiagWin_BottomLToTopR_Middl eLast_Test
		0			
	0	Х			
0	0	Х	Х		
	heckDiagV	Vin(Board	lPosition po	os, char token)	Doggon
Input: pos.getR pos.getC 'token' =	olumn = 2			Output: True State: Unchanged	Reason: This function tests when the winning piece is placed in the middle of a left diagonal to make sure it results correctly.
State: 0	1	2	3		Function Name:
			0		checkDiagWin_BottomRToTopL_Midd leLast_Test
	Х		0		
0	0	Х	Х		
Х	0	X	Х		
hoolean cl	neckDiag\	Vin(Board	Position no	os, char token)	
Input: pos.getR pos.getC 'token' =	ow = 0 olumn = 0		ar osmon po	Output: False State: Unchanged	Reason: This function tests when the board is empty and checks for a diagonal win to ensure it doesn't mark the empty characters as winning.
State: 0	1	2	3		Function Name: checkDiagWin_Empty_Test

boolean checkDiagWin(BoardPosition pos, char token)

boolean c	heckTie()						
Input: State:				Output: True	Reason: This function tests if it accurately states that the board is a tie board		
0	1	2	3	State: Unchanged	when full of pieces		
0	Х	0	0		Function Name:		
Х	0	Х	Х		checkTieWin_Full_Test		
0	Х	0	0				
Х	0	Х	х				
	•	•					
boolean c	heckTie()			1	1		
Input:				Output: False	Reason: This function tests when the board is		
State: 0	1	2	3	State: Unchanged	only populated with blank characters, so it should not result in a tie.		
					Function Name:		
					checkTieWin_Empty_Test		
boolean c	heckTie()			1			
Input:				Output: False	Reason: This function tests when only one		
State: 0	1	2	3	State: Unchanged	piece is placed in the board and then checks for tie, making sure it doesn't		
					see a piece and automatically declare tie without checking.		
					Function Name: checkTieWin_OnlyOne_Test		
					Glicok ric Will_Offiy Offic_Test		
	Х						

Input:				Output: False				Reason:
State:	4	0	2	Otata				This function tests when all pieces but
	1	2	3	State: 0	1	2	3	one slot are filled, and the last spot on the board fulfills another win
0	Х	X		0	Х	Х	0	condition.
0	X	0	X	0	Х	0	Х	Function Name: checkTieWin_finalTokenWin_Test
X	0	X	0	X	0	Х	0	
0	Х	0	Х	0	X	0	X	
					1	1	1	
char whats	sAtPos(Bo	oardPosit	ion pos)	•				
Input: pos.getR pos.getC Token = '	olumn = 0)		Output:				Reason: This function tests checking a position when nothing has been populated
State:				State: Ur	nchanged	I		Function Name: whatsAtPos_Empty_Test
0	1	2	3					
		ļ						
char whats	ΛtDos(R	nardDoeit	ion nos)					_ L
Input:	SALI OS(DI	Jaiui Osii	1011 (103)	Output:				Reason:
pos.getR	ow = 2			'X'				This function tests checking a position
pos.getC Token = 2		2		State: Ur	nchanged	I		when all other positions have been populated
State:								Function Name:
0	<u>1</u>	2	3					whatsAtPos_Full_Test
0	Х	0	0					
Х	0	Х	Х					
0	Х	0	0					
Х	0	Х	X					
-								

char wha	atsAtPos(E	3oardPos	sition pos)		
	Row = 3 Column = - ' '	0	3	Output: ' ' State: Unchanged	Reason: This function tests when the board is partially populated and then testing an empty position to show that it is still empty Function Name:
<u> </u>	<u> </u>	<u> </u>			whatsAtPos_SemiFull_NoPlayer_Test
Х					
0		0	Х		
Х	0	Х	0		
har wha	atsAtPos(E	RoardPos	cition nos)		
Input:		boardi 08	sition pos)	Output: 'X'	Reason:
pos.get	Row = 1 Column = = 'X'	2		State: Unchanged	This function tests when the board is partially populated and then test a full position to show that it has the correct character
0 0	_1	2	3		Function Name:
					whatsAtPos_SemiFull_Occupied_Test
			0		
0		Х	Х		
Х	0	Х	0		
	-1 - A1D /F) ID	::		
<u>:nar wna</u> Input:	atsAtPos(E	<u>soardPos</u>	sition pos)	Output: ' '	Reason:
pos.get pos.get Token =	Row = 3 Column = = ' '	1		State: Unchanged	This function tests when the board is partially populated and then testing an empty position to show that it has the no character
State: 0	1	2	3		Function Name:
0		0	Х		whatsAtPos_SemiFull_Surrounded_T est
Х	Х	0	0		
0	0	Х	X		
Х	0	Х	0		

boolean isPlayerAtPos(BoardPosition pos	s, char player)	
Input: pos.getRow = 0 pos.getColumn = 2 player = X State: 0 1 2 3	Output: True State: Unchanged	Reason: This function tests when the selected character is at the position. Function Name: isPlayerAtPos_IsAt_Test
boolean isPlayerAtPos(BoardPosition po	s, char player)	
Input: pos.getRow = 0 pos.getColumn = 1 player = X State: 0 1 2 3	Output: False State: Unchanged	Reason: This function tests when the selected character is not at the position, but another one is. Function Name: isPlayerAtPos_IsNotAt_Test
boolean isPlayerAtPos(BoardPosition position pos	s, char player)	
Input: pos.getRow = 0 pos.getColumn = 2 player = X	Output: False State: Unchanged	Reason: This function tests when a location that has no piece is checked, but there are other pieces around it.
State: 0 1 2 3		Function Name: isPlayerAtPos_NoPlayer_Test
0 0 X		

Input: pos.getR pos.getC player = 0 State: 0	olumn = 3	2	3 O X O X	Output: True State: Unchanged	Reason: This function tests isPlayerAtPos when a token is placed at the top right of the board Function Name: isPlayerAtPos_topRight_Test
boolean is	PlayerAtF	os(Board	dPosition po	os, char player)	
Input: pos.getRow = 3 pos.getColumn = 1 player = X				Output: False State: Unchanged	Reason: This function tests when the board is full and a player is searched for in a position that they are not in.
State:	1	2	3		Function Name: isPlayerAtPos_Full_Test
О	0	0	Х		, – –
Х	Х	Х	0		
0	0	0	Х		
Х	Х	Х	0		
void dropT	oken(cha	r token, ir	nt column)		
Input: pos.getColumn = 0 'token' = X State:				Output: True State: Unchanged	Reason: This function tests that after a token is dropped in the most left column of an empty board, the left bottom position is the correct token.
0	1	2	3		Function Name: dropToken_Left_Bottom_Test
х					

boolean isPlayerAtPos(BoardPosition pos, char player)

void dropToken(char token, int column)

void drop token (char token, int countri)					
Input: pos.getColumn = 0 'token' = O State:	Output: True State: Unchanged	Reason: This function tests that dropToken can be used to fill the top left of the board and have the correct token.			
0 1 2 3		Function Name:			
0		dropToken_Left_Top_Test			
X					
0					
X					

void dropToken(char token, int column)

Input: pos.getColumn = 1 'token' = X State:				Output: True State: Unchanged	Reason: This function tests to make sure another piece is added on top when the piece is the same rather than forgoing it in the middle of the board.
0	1	2	3		Function Name: dropToken_Middle_of_Board_Test
Х	Х				
Х	0	0			

void dropToken(char token, int column)

Input: pos.getColumn = 3 'token' = X	Output: True	Reason: This function tests to make sure the most right column drops a token in the
State: 0 1 2 3	State: Unchanged	bottom right position of the empty board
		Function Name: dropToken_Right_Bottom_Test
X X		

void dropToken(char token, int column)

void drop tokeri(cri	ai tokeii, ii	it column)			
Input: pos.getColumn = 3 'token' = O State:			Output: True State: Unchanged	Reason: This function tests that dropToken can be used to fill the top right of the board and have the correct token.	
	2	$\frac{3}{0}$		Function Name: dropToken_Right_Top_Test	
		Х			
		0			
		Х			
	•				