GameBoardMem(int rows, int cols, int win) & GameBoard(int rows, int cols, int
win)

Input: Rows=6 Cols=7 Win=4	Output:	Reason:This test case is unique because it tests the default values for gameboard construction Function name: testGameBoardConstructorStandardVals() testGameBoardMemConstructorStandardVals()
Input: Rows=3 Cols=3 Win=3	Output:	Reason: this test case tests the minimum allowable dimensions for gameboard creation Function name: testGameBoardConstructorMinVals() testGameBoardMemConstructorMinVals()
Input: Rows=100 Cols=100 Win=25	Output: 100x100 gameboard	Reason: this test case tests the maximum allowable dimensions for gameboard creation Function name: testGameBoardConstructorMaxVals() testGameBoardMemConstructorMaxVals()

## Boolean checkIfFree(int c)

Input:	Output:	Reason: this test case tests checkiffree on an empty column
State	Checkiffree=tru	Function name:
	е	testGameBoardCheckIfFreeEmptyColumn()
C=0	State of board unchanged	testGameBoardMemCheckIfFreeEmptyColumn()

Input	Output:	Posson: this tost case tests shockiffree on a partially filled
Input:	Output:	Reason: this test case tests checkiffree on a partially filled
State	Checkiffree=tru	column
	е	Function name:
		testGameBoardCheckIfFreePartiallyFilledColumn()
	State of board	testGameBoardMemCheckIfFreePartiallyFilledColumn
	unchanged	()
		\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
X		
C=0		
Input:	Output:	Reason: this test case tests checkiffree on a full column
State	Checkiffree=fals	Function name:
X	е	testGameBoardCheckIfFreeFilledColumn()
X		testGameBoardMemCheckIfFreeFilledColumn()
X	State of board	· ·
X	unchanged	
X		
X		
C=0		

boolean checkHorizWin(BoardPosition pos, char p)

X X X X X  Number to win = 4  Pos.getRow=0  Pos.getColumn=0; P='X'	Output: checkHorizWin=tru e State of board unchanged	Reason: this test case test checkHorizwin if the winning tile is placed on the lefthand side Function name:  testGameBoardCheckHorizWinBottomLeft()  testGameBoardMemCheckHorizWinBottomLeft()
X X X X  Number to win = 4  Pos.getRow=0  Pos.getColumn=3; P='X'	Output: checkHorizWin=tru e State of board unchanged	Reason: this test case test checkHorizwin if the winning tile is placed on the righthand side Function name:  testGameBoardCheckHorizWinBottomRight()  testGameBoardMemCheckHorizWinBottomRight()

	nbe			= 4	Output: checkHorizWin=tru e State of board unchanged	Reason: this test case test checkHorizwin if the winning tile is placed in the middle Function name: testGameBoardCheckHorizWinDropInMiddle() testGameBoardMemCheckHorizWinDropInMiddle())
	.get					
	.get	Colu	ımn	=1;		
P=')	X'				<b>,</b>	
					Output:	Reason: this test case test checkHorizwin if the win
					checkHorizWin=fal	condition is not met
					se	Function name:
					State of board	testGameBoardCheckHorizWinWithBrokenLine()
					unchanged	testGameBoardMemCheckHorizWinWithBrokenLin
Х	Χ	0	Χ	Χ	]	e()
Nur	Number to win = 4			_		
Pos	Pos.getRow=0					
Pos	.get	Colu	ımn	=4;		
P=')	<b>,</b>					

boolean checkVertWin(BoardPosition pos, char p)

					, ,	
				Ш	Output:	Reason: this tests checkvertwin if the winning vertical
					checkVertWin=true	line is on an empty column
		Х			state of board	Function name:
		Х			unchanged	testGameBoardCheckVertWinEmptyColumn()
		Х				testGameBoardCheckVertWinMemEmptyColumn()
		Х				
Nur	nber	to wi	n =	4		
Pos	.getR	ow=3	3			
Pos	.getC	olum	n=3	3		
P=')	ς'					
					Output:	Reason: this tests checkvertwin if the winning vertical
	Х				checkVertWin=true	line is on top of an opponent's piece with room
	Х				state of board	remaining above
	Х				unchanged	Function name:
	Х					testGameBoardCheckVertWinMiddleColumn()
	О					testGameBoardMemCheckVertWinMiddleColumn()
Nur	nber	to wi	in =	4		
Pos						
	_			2		
	P='X'					
Number to win = 4 Pos.getRow=4 Pos.getColumn=2			1			testGameBoardMemCheckVertWinMiddleColumn()

		Χ				Output:	Reason: this tests checkvertwin if the winning line is on
						<b>-   </b>	_
		Χ				checkVertWin=true	top of an opponent's piece with no free spaces above
		Χ				state of board	Function name:
		Χ				unchanged	testGameBoardCheckVertWinTopOfColumn()
		0					testGameBoardMemCheckVertWinTopOfColumn()
		0					
Νι	ımb	er t	0 W	/in	= 4		
Po	s.ge	etRo	w=	5			
	_	etCo			=2		
P=	_		, i di i				
F-	^			1	_	1	
						Output:	Reason: this tests checkvertwin when the win conditions
						checkVertWin=false	have not been met
	Χ					state of board	Function name:
	Х					unchanged	testGameBoardCheckVertWinWithBrokenLine()
	0					71	<pre>testGameBoardMemCheckVertWinWithBrokenLine()</pre>
	Х					7	
Νι	Number to win = 4		= 4	-			
Ро	Pos.getRow=3			3			
Po	s.ge	etCo	lun	nn	=1		
Pos.getColumn=1 P='X'				1			

boolean checkDiagWin(BoardPosition pos, char p)

X	Output: checkDiagWin= true state of board unchanged	Reason: this test checkdiagwin with an ascending line where the winning token is placed in the tip right Function name testGameBoardCheckDiagWinAscendingLastTokenTop Right() testGameBoardMemCheckDiagWinAscendingLastToken TopRight()
X O O O Number to win = 4 Pos.getRow=0 Pos.getColumn=0 P='X'	Output: checkDiagWin= true state of board unchanged	Reason: this test checkdiagwin with an ascending line where the winning token is placed in the bottom left Function name testGameBoardCheckDiagWinAscendingLastTokenBot tomLeft() testGameBoardMemCheckDiagWinAscendingLastToken BottomLeft()

	,	-			1	
					Output:	Reason: this test checkdiagwin with an ascending line
					checkDiagWin=	where the winning token is placed in the middle
			Х		true	Function name
		Х	0		state of board	testGameBoardCheckDiagWinAscendingLastTokenMid
	Χ	0	0		unchanged	dle()
Х	0	0	0			testGameBoardMemCheckDiagWinAscendingLastToken
		r to v	win	= 4		Middle()
		Row		-		1.1.3.1.1.1
	_	Colu		=1		
P='	_	Colu		_		
					Output:	Reason: this test checkdiagwin with an ascending line
					checkDiagWin=	where the win conditions are not met
			V		false	Function name
			X		state of board	
			0			testGameBoardCheckDiagWinAscendingBrokenLine()
	Х		О		unchanged	testGameBoardMemCheckDiagWinAscendingBrokenLin
X			0			e()
		r to ۱		= 4		
Pos	.get	Row:	=3			
Pos	.get	Colu	mn:	=3		
P='	Χ'					
					Output:	Reason: this test checkdiagwin with a descending line
					checkDiagWin=	where the winning token is placed top left
Х					true	Function name
О	Х				state of board	testGameBoardCheckDiagWinDescendingLastTokenTo
0	0	Х			unchanged	pLeft()
0	0	t t	Х			testGameBoardMemCheckDiagWinDescendingLastToke
	1 -	r to v		- 1		nTopLeft()
		Row:		- 4		mople ()
	_	Colu		_1		
P='	_	Colu	11111	-1		
F - 4	<del>^</del>		- 1		O. stravita	Descent this test shoot should a with a descending line
					Output:	Reason: this test checkdiagwin with a descending line
1					checkDiagWin=	where the winning token is placed in the bottom right
X					true	Function name
0	Х				state of board	testGameBoardCheckDiagWinDescendingLastTokenBo
0	0	Х			unchanged	ttomRight()
0	0	0	Х			testGameBoardMemCheckDiagWinDescendingLastToke
Nur	mbe	r to ۱	win	= 4		nBottomRight()
Pos	.get	Row:	=3			
Pos	.get	Colu	mn:	=0		
P='	_					
Ь					1	I .

	Output:	Reason: this test checkdiagwin with a descending line
	checkDiagWin=	where the winning token is placed in the middle
X	true	Function name
O X	state of board	testGameBoardCheckDiagWinDescendingLastTokenMi
0 0 X	unchanged	ddle()
0 0 0 X		testGameBoardMemCheckDiagWinDescendingLastToke
Number to win = 4		nMiddle()
Pos.getRow=0		
Pos.getColumn=3		
P='X'		

## Boolean checkTie()

0	Х	0	Output:	Reason: this tests checkTie when the board is full
Х	0	Х	checkTie()=true	Function name
Х	0	Х	state of board	<pre>testGameBoardCheckTieFullBoard()</pre>
	•		unchanged	<pre>testGameBoardMemCheckTieFullBoard()</pre>
0	Х		Output:	Reason: this tests checkTie when there is one free
Χ	0	Х	checkTie()=false	space remaining
Х	0	Х	state of board	Function name
	•		unchanged	<pre>testGameBoardCheckTieOneFreeSpace()</pre>
				<pre>testGameBoardMemCheckTieOneFreeSpace()</pre>
			Output:	Reason: this tests checkTie when the board is
			checkTie()=false	empty
			state of board	Function name
		<u> </u>	unchanged	<pre>testGameBoardCheckTieEmptyBoard()</pre>
				<pre>testGameBoardMemCheckTieEmptyBoard()</pre>
0	Х		Output:	Reason: this tests checkTie when there is one free
X	0		checkTie()=false	column remaining
Х	0		state of board	Function name
			unchanged	<pre>testGameBoardCheckTieOneFreeColumn()</pre>
				testGameBoardMemCheckTieOneFreeColumn()

#### char whatsAtPos(BoardPosition pos)

	Output:	Reason: this tests whatsatpos on an empty board
	whatsAtPos(pos)	Function name
	=' '	testGameBoardWhatsAtPosEmptyBoard()
	State of board	testGameBoardMemWhatsAtPosEmptyBoard()
	unchanged	
Pos.getRow=0		
Pos.getColumn=0		

X Pos.getRow=0 Pos.getColumn=0	Output: whatsAtPos(pos) ='X' State of board unchanged	Reason: this tests whatsatpos on the bottom of a column with one token played Function name testGameBoardWhatsAtPosBottomOfColumnOneTokenPl ayed() testGameBoardMemWhatsAtPosBottomOfColumnOneToke nPlayed()
O	Output: whatsAtPos(pos) ='O' State of board unchanged	Reason: this tests whatsatpos on the top of a full column Function name testGameBoardWhatsAtPosTopOfColumnFullColumn() testGameBoardMemWhatsAtPosTopOfColumnFullColumn()
O X O X O X O X O X O X O X O X O X O X	Output: whatsAtPos(pos) ='X' State of board unchanged	Reason: this tests whatsatpos on the middle of a full column Function name testGameBoardWhatsAtPosMiddleOfColumnFullColumn () testGameBoardMemWhatsAtPosMiddleOfColumnFullColumn()
X O X O X O X O SetRow=4 Pos.getColumn=0	Output: whatsAtPos(pos) ='X' State of board unchanged	Reason: this tests whatsatpos on the highest occupied space of a partially filled column Function name testGameBoardWhatsAtPosTopOfPartiallyFullColumn () testGameBoardMemWhatsAtPosTopOfPartiallyFullColumn()

# Boolean isPlayerAtPos(BoardPosition pos, char player)

	Output:	Reason: this tests if player is at position for an unoccupied
	isPlayerAtPos(pos	space at the bottom of a column
	,'X')=false	Function name
	state of board	testGameBoardIsPlayerAtPosEmptySpace()
	unchanged	testGameBoardMemIsPlayerAtPosEmptySpace()
Pos.getRow=0		
Pos.getColumn=0		
Player='X'		

X Pos.getRow=0 Pos.getColumn=0 Player='X'	Output: isPlayerAtPos(pos ,'X')=true state of board unchanged	Reason: this tests if the correct player is at position for an occupied space at the bottom of a column Function name testGameBoardIsPlayerAtPosOccupiedSpaceCorrectPlayer() testGameBoardMemIsPlayerAtPosOccupiedSpaceCorrectPlayer()
X Pos.getRow=0 Pos.getColumn=0 Player='0'	Output: isPlayerAtPos(pos ,'O')=false state of board unchanged	Reason: this tests if the incorrect player is at the position of the occupied space at the bottom of a column Function name testGameBoardIsPlayerAtPosOccupiedSpaceIncorrectPlayerBottomOfColumn() testGameBoardMemIsPlayerAtPosOccupiedSpaceIncorrectPlayerBottomOfColumn()
O X O X O O O O O O O O O O O O O O O O	Output: isPlayerAtPos(pos ,'O')=true state of board unchanged	Reason: this tests if the correct player is at the position of the occupied space at the top of a column Function name testGameBoardIsPlayerAtPosOccupiedSpaceTopOfColumn() testGameBoardMemIsPlayerAtPosOccupiedSpaceTopOfColumn()
O X O X O O O O O O O O O O O O O O O O	Output: isPlayerAtPos(pos ,'X')=false state of board unchanged	Reason: this tests if the correct player is at the position of the occupied space at the middle of a column Function name testGameBoardIsPlayerAtPosOccupiedSpaceMiddleOf Column() testGameBoardMemIsPlayerAtPosOccupiedSpaceMiddleOfColumn()

# void dropToken(char p, int c)

	Output					Reason: this tests using droptoken on an empty
						column
						Function name
						testGameBoardDroptokenEmptyColumn()
						testGameBoardMemDroptokenEmptyColumn()
						ces coameboar diffembl opcokerizing cycordiniir()
			<u> </u>			
P = 'X'	Х					
C = 0	State					
						Reason: this tests using droptoken to partially fill up a
						column
						Function name
	Х					testGameBoardDroptokenPartialColumn()
	0					testGameBoardMemDroptokenPartialColumn()
	X			-		ces esameboar ariembri op concern ar etateotamin()
				1		
P = 'X','O','X'	State					
C = 0,0,0			T	1	1 1	
	Х					Reason: this tests using droptoken to completely fill
	Х					up a column
	X					Function name
	Х					testGameBoardDroptokenFullColumn()
	Х					testGameBoardMemDroptokenFullColumn()
	Х					' '
P = 'X', 'X', 'X', 'X',	State		<u> </u>	<u> </u>		
'X', 'X'	State					
C = 0,0,0,0,0,0						
C = 0,0,0,0,0,0			1			December this tests using drawfalcon to completely fill
						Reason: this tests using droptoken to completely fill
						up a row
						Function name
						testGameBoardDroptokenFullRow()
						testGameBoardMemDroptokenFullRow()
	ХХ	ХХ	Χ	Χ	Х	
P = 'X', 'X', 'X', 'X',	State	l l			<u> </u>	
'X', 'X', 'X'	0 10.10					
C = 0,0,0,0,0,0,0						
C = 0,0,0,0,0,0,0		V		^		Reason: this tests using droptoken to completely fill
	0	X		0		• , , , ,
	X	0		Χ		up a board
	X O X			Function name		
P = 'X', 'X', 'O', 'O',	State					testGameBoardDroptokenFillEntireBoard()
'O', 'X', 'X', 'X', 'O'						testGameBoardMemDroptokenFillEntireBoard(
C = 0, 0, 0, 1, 1,						
1, 2, 2, 2						'
. , ,						