Project 4 Test Cases (GameBoard & GameBoardMem)

Constructor(int inRow, int inColumn, int inNumToWin)

Input:	Output:	Reason:
inRow = 3 inColumn = 3 inNumToWin = 3	State: (number to win = 3) BOARD IS 3x3 and empty	This test is unique because it tests that the constructor is able to correctly initialize an empty board to the minimum dimensions possible.
		Function Name: testConstructor_ 3_3_3_min_dimensions

Constructor(int inRow, int inColumn, int inNumToWin)

Input:	Output:	Reason:
inRow = 100 inColumn = 100 inNumToWin = 25	State: (number to win = 25)	This test is unique because it tests that the constructor is able to correctly initialize an empty board to the maximum dimensions possible.
	BOARD IS 100x100 and empty	Function Name: testConstructor_ 100_100_25_max_ dimensions

Constructor(int inRow, int inColumn, int inNumToWin)

Input:	Output:	Reason:
		This test is unique
inRow = 9	State: (number to win $= 5$)	because it tests that
inColumn = 7		the constructor can
		correctly initialize
inNumToWin = 5		an empty board to
		dimensions containing
		different # of cols
		and rows
	BOARD IS 9x7 and empty	
	BOARD IS 3x7 and empty	Function Name:
		testConstructor 9 7 5
		diff_dimensions

boolean checkIfFree(int c)

Inpu	Input:				Output:	Reason:
r					checkIfFree = true	This test is unique
State:						because the left most
	u v w x			X	state of the board is unchanged	column is almost full and all other columns
p	p q r s t			t	unchanged	are full.
k	1	m	n	o		
f	g	h	i	j		Function Name:
a	a b c d e		e		testCheckIFFree_	
c = 0						board_full_except_ left_column

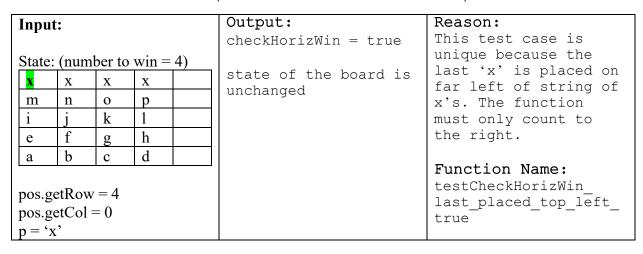
boolean checkIfFree(int c)

Input:	Output:	Reason:
_	<pre>checkIfFree = false</pre>	This test is unique
State:		because the column is
l e	beace of the Routa is	full and no other
d	unchanged	columns are almost
		full.
b		Function Name:
a		testCheckIfFree_
		column_full_false
c = 4		

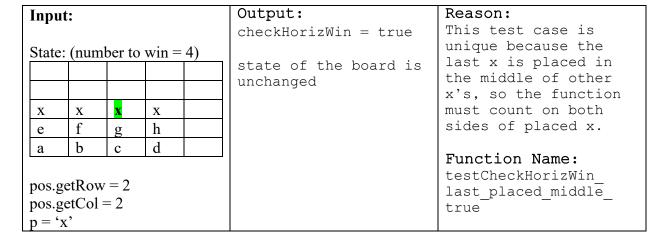
boolean checkIfFree(int c)

Inpu	Input:				Output:	Reason:
•					checkIfFree = true	This test is unique
State	State:					because the right
u	v	W	X		state of the board is	most column is almost
u	_ ·	VV	Λ		unchanged	full and all other
p	q	r	S	t		columns are full.
k	1	m	n	o		
f	g	h	i	j		Function Name:
a	b	С	d	e		testCheckIfFree_
c = 4	c = 4					board_full_except_ right_column

boolean checkHorizWin(BoardPosition pos, char p)



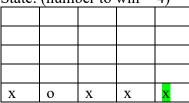
boolean checkHorizWin (BoardPosition pos, char p)



boolean checkHorizWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)



pos.getRow = 0pos.getCol = 4p = 'x'

Output:

checkHorizWin = false

state of the board is unchanged

This test is unique because the last x is placed on the righthand side and function must count only to the left, until it runs into a 'o' character.

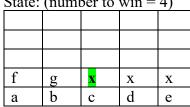
Function Name:

testCheckHorizWin last placed bottom right false

boolean checkHorizWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)



pos.getRow = 1pos.getCol = 2p = 'x'

Output:

checkHorizWin = false

state of the board is unchanged

Reason:

This test is unique because the last 'x' is placed on left-hand side of string of x's and the function must count only to the right until it hits end of board.

Function Name:

testCheckHorizWin last placed left false

boolean checkVertWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

State. (Hullidel to will – 4)						
		X				
		X				
		X				
		X				
		0				

pos.getRow = 4pos.getCol = 2p = 'x'

Output:

checkVertWin = true

state of the board is unchanged

Reason:

This test is unique because there is an 'o' below string of x's and final 'x' is on top row.

Function Name:

testCheckVertWin middle last placed in top row true

boolean checkVertWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

X		
X		
X		
X		

pos.getRow = 3 pos.getCol = 0 p = 'x'

Output:

checkVertWin = true

state of the board is unchanged

Reason:

This test is unique because the last 'x' is not placed in top row of board and the string of x's are located at left-side of board.

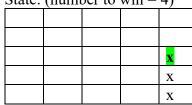
Function Name:

testCheckVertWin_left_ last_not_in_top_row_ true

boolean checkVertWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)



pos.getRow = 2 pos.getCol = 4 p = 'x'

Output:

checkVertWin = true

state of the board is unchanged

Reason:

This test is unique because the string of x's are located at right-side of the board and the function must count down until reaching bottom of board.

Function Name:

testCheckVertWin_right
_false

boolean checkVertWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

Diate.	(1101111	JOI TO	* * 111	• ,
X				
X				
X				
0				
X				

pos.getRow = 4pos.getCol = 0

pos.get

Output:

checkVertWin = false

state of the board is unchanged

Reason:

This test is unique because the function must count down from last 'x' placed until it hits an 'o' character and last x is placed in top row.

Function Name:

testCheckVertWin_left_ last_placed_in_top_row _false boolean checkDiagWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

State. (number to win - 4)						
			X			
		X	f			
	X	d	e			
X	a	b	c			

Output:

checkDiagWin = true

state of the board is unchanged

Reason:

This test is unique because the last token is placed in the middle of the string of x's so the function must count in both directions and it is testing positive diagonal.

Function Name:

testCheckDiagWin_last_
placed_in_middle_pos_
diag

boolean checkDiagWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

State. (Hamber to will 1)						
	X					
	f	X				
	e	d	X			
	c	b	a	X		

pos.getRow = 2 pos.getCol = 2 p = 'x'

Output:

checkDiagWin = true

state of the board is unchanged

Reason:

This test is unique because the last token is placed in the middle of the string of x's so the function must count in both directions and it is testing negative diagonal.

Function Name:

testCheckDiagWin_last_
placed_in_middle_neg_
diag

boolean checkDiagWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

State. (Halliber to will +)							
				X			
			X	k			
		X	i	j			
	X	f	g	h			
a	b	С	d	e			

Output:

checkDiagWin = true

state of the board is unchanged

This test is unique because the last token is placed in top right and is testing the positive diagonal. The function must count down and to the left.

Function Name:

testCheckDiagWin last last placed in top right

boolean checkDiagWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

	_			
X				
k	X			
j	i	X		
h	g	f	X	
e	d	c	b	a

Output:

checkDiagWin = true

state of the board is unchanged

Reason:

This test is unique because the last token is placed in top left and it tests the negative diagonal. The function must count down and to the right

Function Name:

testCheckDiagWin last placed in top left

boolean checkDiagWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

State. (Hullioet to will 4)									
			X						
		X	f						
	X	d	e						
X	a	b	c						

pos.getCol = 0p = 'x'

Output:

checkDiagWin = true

state of the board is unchanged

Reason:

This test is unique because the last token is placed in bottom left, and it tests the positive diagonal. The function must count up and to the right.

Function Name:

testCheckDiagWin last placed in bottom left

boolean checkDiagWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

State. (Hullioci to will – 4)									
	X								
	f	X							
	e	d	X						
	c	b	a	X					

Output:

checkDiagWin = true

state of the board is unchanged

Reason:

This test is unique because the last token is placed in the bottom right and it is testing the negative diagonal. The function must count up and to the left.

Function Name:

testCheckDiagWin_last
_placed_in_bottom_
right

boolean checkDiagWin(BoardPosition pos, char p)

Input:

State: (number to win = 4)

X				
1	X			
k	j	X		
i	h	g	f	
e	d	c	b	a

pos.getRow = 4 pos.getCol = 0 p = 'x'

Output:

checkDiagWin = false

state of the board is unchanged

Reason:

This test is unique because the last token is placed in top left and the function must count down and to the right until an 'o' is next.

Function Name:

testCheckDiagWin_last_
placed_in_top_left_
false

boolean checkTie()

Inpu	t:				Output:	Reason:
-					<pre>checkTie = false</pre>	This test is unique
State						because the entire
State	•	1	1	1	state of the board is	board is full except
	u	V	W	X	unchanged	for the top left
р	q	r	S	t	differialiged	spot.
k	1	m	n	o		
f	g	h	i	j		Function Name:
a	b	c	d	e		testCheckTie_full_
			•			except_top_left_false

boolean checkTie()

Inpu	Input:				Output:	Reason:
_					<pre>checkTie = false</pre>	This test is unique
State						because the entire board
u	v	W	X		state of the board	is full except for the
р	q	r	S	t	is unchanged	top right spot.
k	1	m	n	0		Function Name:
f	g	h	i	j		testCheckTie_full_except_
a	b	c	d	e		top_right_false

boolean checkTie()

Input	t :				Output:	Reason:
•					<pre>checkTie = false</pre>	This test is unique
State:						because no columns in
State.					state of the board is	the board are full
					unchanged	and the board is not
						almost full.
f	g	h	i	j		Function Name:
a	b	c	d	e		testCheckTie_no_full_
	•	•	•	· ·		columns_false

boolean checkTie()

Inpu	it:				Output: checkTie = true	Reason: This test is unique
State	:					because all columns are
u	V	W	X	у	state of the board is unchanged	full.
p	q	r	S	t	diffialiged	Function Name:
k	1	m	n	o		testCheckTie full board
f	g	h	i	j		_true
a	b	c	d	e		

char whatsAtPos(BoardPosition pos)

Input: State:					Output: whatsAtPos = 'a'	Reason: This test is unique because it is testing
u	v	W	X	у	state of the board is unchanged	that whatsAtPos can properly return
р	q	r	S	t	diferialiged	character in border
k	1	m	n	О		case where char is at
f	g	h	i	j		bottom left corner of
a	b	c	d	e		board.
	getRov getCol					Function Name: testWhatsAtPos_char_ at_bottom_left_corner

char whatsAtPos(BoardPosition pos)

I	nput	•				Output:	Reason:
						whatsAtPos = 'u'	This test is unique
S	State:						because it is testing
۱ĭ	<u></u>	v	w	X	V	state of the board is	that whatsAtPos can
	u	V	VV	Λ	y	unchanged	properly return
	p q r s t				t		character in border
	k	1	m	n	o		case where char is at
	f	g	h	i	i		top left corner of
	a	b	С	d	e		board.
		-4D	- 4				Function Name:
		etRow					testWhatsAtPos char
p	os.ge	etCol	= 0				at_top_left_corner

char whatsAtPos(BoardPosition pos)

Input:											
State:											
u	V	W	X	у							
p	q	r	S	t							
k	1	m	n	О							
f	g	h	i	j							
a	b	c	d	e							

pos.getRow = 0 pos.getCol = 4

Output:

whatsAtPos = 'e'

state of the board is unchanged

Reason:

This test is unique because it is testing that whatsAtPos can properly return character in border case where char is at bottom right corner of board.

Function Name:
testWhatsAtPos_char_
at bottom right corner

char whatsAtPos(BoardPosition pos)

Input:

State:

State.									
u	V	W	X	y					
p	q	r	S	t					
k	1	m	n	o					
f	g	h	i	j					
a	b	c	d	e					

pos.getRow = 4 pos.getCol = 4

Output:

whatsAtPos = 'y'

state of the board is unchanged

Reason:

This test is unique because it is testing that whatsAtPos can properly return character in border case where char is at top right corner of board.

Function Name:

testWhatsAtPos_char_ at_top_right_corner

char whatsAtPos(BoardPosition pos)

Input:

k l m n f g h i j a b c d e

pos.getRow = 2 pos.getCol = 2

Output:

whatsAtPos = ' '

state of the board is unchanged

Reason:

This test is unique because it is testing to make sure whatsAtPos properly returns a space character when board space is empty.

Function Name:

testWhatsAtPos_return
_space

boolean isPlayerAtPos(BoardPosition pos, char player)

Input:

State:

State.						
u	v	W	X	y		
p	q	r	S	t		
k	1	m	n	0		
f	g	h	i	j		
a	Ъ	c	d	e		

Output:

isPlayerAtPos = true

state of the board is unchanged

Reason:

This test is unique because it is testing that isPlayerAtPos can correctly identify a char in the border case that the char is in bottom left corner of board.

Function Name:

testIsPlayerAtPos_
char_at_bottom_left_
corner

boolean isPlayerAtPos(BoardPosition pos, char player)

Input:

State:

u	v	W	X	у	
р	q	r	S	t	
k	1	m	n	О	
f	g	h	i	j	
a	ь	С	d	е	

pos.getRow = 4 pos.getCol = 0 player = 'u'

Output:

isPlayerAtPos = true

state of the board is unchanged

Reason:

This test is unique because it is testing that isPlayerAtPos can correctly identify a char in the border case that the char is in top left corner of board.

Function Name:

testIsPlayerAtPos_
char_at_top_left_
corner

boolean isPlayerAtPos(BoardPosition pos, char player)

Inj	Input:					<pre>Output: isPlayerAtPos = true</pre>	Reason: This test is unique
State:							because it is testing
u	u v w x y		у	state of the board is unchanged	that isPlayerAtPos can correctly		
р	p q r s t k l m n o f g h i j		t	9	identify a char in		
k			o		the border case that		
f			j		the char is in the		
a	b)	c	d	e		bottom right corner of board.
						or board.	
pos.getRow = 0							Function Name:
pos.getCol = 4						testIsPlayerAtPos	
pla	player = 'e'						char_at_bottom_right_

boolean isPlayerAtPos(BoardPosition pos, char player)

Input:					Output:	Reason:
					isPlayerAtPos = true	This test is unique
State	•					because it is testing
u	$\frac{\cdot}{\mathbf{v}}$	w	X	w.	state of the board is	that isPlayerAtPos
u	v	VV	Λ	<u>y</u>	unchanged	can correctly
p	p q r s t			t		identify a char in
k	k 1 m n o		o		the border case that	
f	f g h i j		i		the char is in the	
a b c d e		e		top right corner of		
						the board.
pos.getRow = 4 pos.getCol = 4 player = 'y'						Function Name: testIsPlayerAtPos_ char_at_top_right_ corner

corner

boolean isPlayerAtPos(BoardPosition pos, char player)

pos.getRow = 2 pos.getCol = 2 player = 'x'

Output:

isPlayerAtPos = false

state of the board is unchanged

Reason:

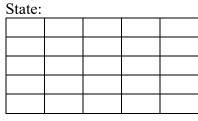
This test is unique because it is testing that isPlayerAtPos can correctly identify if a char is not at the specified location.

Function Name:

testIsPlayerAtPos_
middle_of_board_false

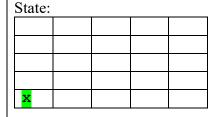
void dropToken(char p, int c)





p	=	' x'
c	=	0

Output:



Reason:

This test is unique because it is testing the border case of dropping a token into bottom left corner of board.

Function Name:

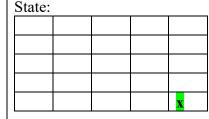
testDropToken_empty_
board_bottom_left

void dropToken(char p, int c)

State:

p = x, c = 4

Output:



Reason:

This test is unique because it is testing the border case of dropping a token into bottom right corner of board.

Function Name:

testDropToken_empty_ board_bottom_right void dropToken(char p, int c)

Input:	Output:				
State:			State:		
			e		
d			d		
С			С		
b			b		
a			a		

Reason:

This test is unique because it is testing the border case of dropping a token into top left corner of the board.

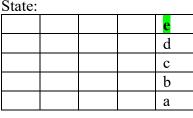
Function Name:

testDropToken top left _corner

void dropToken(char p, int c)

Input:		Outpu
State:		State:
	d	
	c	
	b	
	a	
	_	
p = 'e'		

Output	:



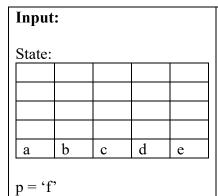
Reason:

This test is unique because it is testing the border case of dropping a token into the top right corner of the board.

Function Name:

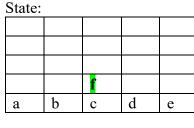
testDropToken_top_ right corner

void dropToken(char p, int c)



c = 4

c = 2



Reason:

This test is unique because it is testing if dropToken properly places new token on top of already existing tokens.

Function Name:

testDropToken middle column not empty