CheckerBoard(int aDimension) – testCheckerBoard_Con_10x10

Ou	tput	:								
Sta	te:		_				_			
x	*	х	*	x	*	х	*	x	*	
*	х	*	х	*	х	*	х	*	х	
x	*	х	*	х	*	х	*	х	*	
*	х	*	х	*	х	*	х	*	х	
	*		*		*		*		*	
*		*		*		*		*		
О	*	0	*	0	*	0	*	0	*	
*	0	*	0	*	0	*	0	*	o	
О	*	0	*	0	*	0	*	0	*	
*	o	*	o	*	o	*	o	*	o	
								·		
	* * * * 0	State: x * x	X * X * X * X * X * X * * X * * X * * O * O * O * O * O	State: X * X * * X * X * X * * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * X * * X * X * X * X * X * X * X * X * X * X * X * X * X * * X * * X * X * X * * X * X *	State: X	State: X	State: X	State: X	State: x * x * x * x * x * x * x * x * x * x * x	State: x * x * x * x * x * x * x * x * x<

CheckerBoard(int aDimension) – testCheckerBoard_Con_8x8

Input:aDimension = 8	Out	put:						
State:N/A	State	e:						
	х	*	х	*	х	*	х	*
	*	х	*	х	*	х	*	х
	х	*	х	*	х	*	х	*
	*		*		*		*	
		*		*		*		*
	*	О	*	0	*	0	*	0
	0	*	0	*	0	*	0	*

*	0	*	0	*	0	*	0	

CheckerBoard(int aDimension) – testCheckerBoard_Con_16x16

Input:aDimension = 16	Out	tput:															
State:N/A	Stat	e:															
	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	
	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	
	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	
	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	
	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	
	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	
	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	
	*		*		*		*		*		*		*		*		
		*		*		*		*		*		*		*		*	
	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	
	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	
	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	
	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	
	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	
	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	
	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	
		•			•			•								•	

whatsAtPos(BoardPosition pos) – testWhatsAtPos_1_1_x

•				1	
	n	n	ıı	т	-
		\sim	u	ι	

pos = (1, 1)

State:

х	*	х	*	х	*	x	*
*	X	*	x	*	Х	*	Х
Х	*	х	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: 'x'

State:

[state of the board is unchanged]

 $whats At Pos (Board Position\ pos) - test Whats At Pos_min Row_2_x$

Input:

pos = (0, 2)

State:

Output: 'x'

State: [state of the board is unchanged]

 x
 *
 x
 *
 x
 *
 x
 *

 *
 x
 *
 x
 *
 x
 *
 x

Х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	О	*	0	*	О	*
*	0	*	0	*	0	*	О

 $whats At Pos (Board Position\ pos) - test Whats At Pos_3_max Column$

Inpu	t:							Output: ' '
pos	= (3,	7)						
State	e:							State: [state of the board is unchanged]
х	*	х	*	х	*	х	*	
*	х	*	х	*	х	*	х	
х	*	х	*	х	*	х	*	
*		*		*		*		
	*		*		*		*	
*	0	*	0	*	0	*	0	
0	*	0	*	0	*	0	*	
*	o	*	0	*	0	*	0	
		I				I		

whatsAtPos(BoardPosition pos) – testWhatsAtPos_maxRow_6_*

Input	t:							Output: '*'
pos	= (7,	6)						
State	e:				•			State: [state of the board is unchanged]
х	*	х	*	х	*	х	*	
*	х	*	х	*	x	*	х	
х	*	х	*	х	*	х	*	
*		*		*		*		
	*		*		*		*	
*	0	*	0	*	0	*	0	
О	*	0	*	О	*	0	*	
*	О	*	0	*	0	*	0	
		•	•	•	•	•	•	

whatsAtPos(BoardPosition pos) – testWhatsAtPos_6_minCol_o

Input	:							Output:
pos	= (6,0	0)						0
State	:	•			•			State: [state of the board is unchanged]
x	*	х	*	х	*	х	*	
*	Х	*	x	*	х	*	х	
х	*	x	*	х	*	х	*	
*		*		*		*		
	*		*		*		*	
*	0	*	0	*	0	*	0	
O	*	0	*	0	*	О	*	
*	0	*	0	*	0	*	О	

placePiece(BoardPosition pos, char player) – testPlacePiece_6_6_x

Input	:															
pos =	= (6, 6	3)							Outp	ut: N	/A					
playe	er = 'x	('							State	e:						
State	:															
х	*	х	*	х	*	х	*		х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х		*	x	*	x	*	х	*	х
х	*	х	*	х	*	х	*		х	*	х	*	х	*	х	*
*		*		*		*			*		*		*		*	
	*		*		*		*			*		*		*		*
*	0	*	0	*	О	*	0		*	0	*	0	*	0	*	0
О	*	0	*	0	*	o	*		0	*	0	*	0	*	x	*
*	0	*	0	*	0	*	О		*	0	*	0	*	0	*	О
								'								

placePiece(BoardPosition pos, char player) – testPlacePiece_4_1_o

Input	:														
pos =	= (4,	1)						Outp	ut: N	/A					
playe	er = 'd	o'						State	e:						
State): 														
х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*
*	x	*	x	*	x	*	х	*	x	*	x	*	x	*	х
х	*	х	*	х	*	х	*	х	*	х	*	x	*	х	*
*		*		*		*		*		*		*		*	
	*		*		*		*		o		*		*		*
*	0	*	0	*	0	*	О	*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	О	*	0	*	0	*	0	*	0

 $place Piece (Board Position\ pos,\ char\ player) - test Place Piece _minRow _5_x$

Input	::														
pos =	= (0,	5)						Outp	ut: N	/A					
playe	er = 'x	ς'						State	e:						
State	e:	_							_						
х	*	х	*	х	*	х	*	х	*	x	*	x	×	x	*
*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	*
*		*		*		*		*		*		*		*	
	*		*		*		*		*		*		*		*
*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0
О	*	0	*	o	*	0	*	0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0

placePiece(BoardPosition pos, char player) – testPlacePiece_2_minColumn_o

Input	::							Out	put: N	N/A					
pos =	= (2,	0)													
playe	er = 'o	0'													
State	e:							State	e:						
x	*	х	*	x	*	х	*	x	*	х	*	x	x	x	*
*	х	*	х	*	x	*	х	*	x	*	x	*	x	*	x
X	*	х	*	х	*	х	*	O	*	х	*	х	*	х	*
*		*		*		*		*		*		*		*	
	*		*		*		*		*		*		*		*
*	0	*	О	*	0	*	0	*	0	*	0	*	0	*	0
0	*	0	*	0	*	О	*	0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0
	•	•		•	•	•			•		•		•	•	

placePiece(BoardPosition pos, char player) – testPlacePiece_4_maxColumn_x

nput:	•														
oos =	(4,	7)						Outp	out: N	I/A					
olaye	r = 'x	('													
State	:							State	e:						
х	*	х	*	х	*	х	*								
*	Х	*	х	*	х	*	х	х	*	х	*	х	х	х	*
Х	*	х	*	х	*	х	*	*	х	*	х	*	х	*	х
*		*		*		*		х	*	х	*	х	*	х	*
	*		*		*		*	*		*		*		*	
*	0	*	0	*	0	*	О		*		*		*		X
0	*	0	*	0	*	0	*	*	0	*	0	*	0	*	0
*	0	*	0	*	0	*	О	0	*	0	*	0	*	0	*
								*	0	*	0	*	0	*	0

 $place Piece (Board Position\ pos,\ char\ player) - test Place Piece _maxRow _2_x$

Input	:														
pos =	= (7, 2	2)						Outp	ut: N	/A					
playe	er = 'c	o'						State	e:						
State):														
x	*	х	*	х	*	х	*	x	*	x	*	x	x	x	*
*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*	х	*	х	*	х	*	x	*
*		*		*		*		*		*		*		*	
	*		*		*		*		*		*		*		*
*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	О	*	0	o	0	*	0	*	О

getPieceCounts(void) - testGetPieceCounts_x_6_o_min

Input:	N/A
--------	-----

Output:

 $pieceCount = \{('x', 6), ('o', 0)\}$

State:

pieceCount = $\{('x', 6), ('o', 0)\}$

	*		*		*		*
*		*		*	X	*	X
×	*	×	*	×	*	×	*
*		*		*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

State:

pieceCount = [pieceCount hashmap is unchanged]

[state of board is unchanged]

getPieceCounts(void) - testGetPieceCounts_x_max_o_3

Input: N/A

Output:

pieceCount = $\{('x', 12), ('o', 8)\}$

State:

pieceCount = {('x', 12), ('o', 8)}

State:

pieceCount = [pieceCount hashmap is unchanged]

<mark>x</mark>	*	X	*	x	*	x	*
*	X	*	×	*	×	*	x
×	*	x	*	×	*	x	*
*		*		*		*	
	*		*		*		*
*	O	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*		*		*		*	
*		*		*		*	

 $get Viable Directions (void) - test Get Viable Directions_PLAYER_ONE$

Input:	Output:
None	viableDirections = {'NE','NW','SE', 'SW'}
State:	
x = PLAYER_ONE	State:
viableDirections = {'NE','NW', 'SE', 'SW'}	

 $add Viable Directions (char player, Direction Enum dir) - test Add Viable Directions _PLAYER_ONE$

Input:	Output:
player = 'x'	viableDirections = NW
dir = NW	State:
State:	
x = PLAYER_ONE	
viableDirections = null	

$getRowNum(void) - getRowNumTest_8$

Input	t: non	ie						Output: 8
								State:
State	e: 8x8	B boa	rd					[state of the board is unchanged]
х	*	х	*	х	*	х	*	
*	x	*	x	*	x	*	x	
х	*	х	*	х	*	х	*	
*		*		*		*		
	*		*		*		*	
*	0	*	0	*	o	*	О	
0	*	0	*	0	*	х	*	
*	О	*	0	*	0	*		

getCo	olNun	n(void	d) – g	etCo	lNum	Test_	_8	
Inpu	t:							Output: 8
State	e: 8x	8 boa	ırd					State: [state of the board is unchanged]
x	*	х	*	х	*	х	*	
*	х	*	х	*	х	*	х	
x	*	х	*	х	*	х	*	
*		*		*		*		
	*		*		*		*	
*	0	*	0	*	0	*	0	
0	*	О	*	О	*	х	*	
*	0	*	0	*	О	*		

checkPlayerWin(Character player) – testCheckPlayerWin_PLAYER_ONE_Win

_										T
	Input	·.								Output:
										True
	State) :								State:
						,	,		1	
	Х	*	х	*	х	*	х	*		
	*	х	*	x	*	х	*	х		
	х	*	х	*	х	*	х	*		
	*		*		*		*			
		*		*		*		*		
	*		*		*		*			
		*		*		*		*		
	*		*		*		*			
	piece	Cou	nt = {	'('x', '	12), ('	o', 0)}	<u> </u>	4	
_				<u> </u>	<i>,,</i> (•			

 $check Player Win (Character\ player) - test Check Player Win _PLAYER _TWO _NoW in$

Input	:							Output:
State	:							False
								State:
х	*		*		*		*	
*		*		*		*		
	*		*		*		*	
*		*		*		*		
	*		*		*		*	

 $crown Piece (Board Position\ pos Of Player) - test Crown Piece _MinRow _MinColumn _x$

Input: posOfPlayer = (0, 0)

State: 8x8 board

Otato	. 0/10	Doai	<u> </u>				
х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
О	*	О	*	0	*	О	*
*	0	*	О	*	0	*	

Output: N/A

State: 8x8 board

X	*	х	*	х	*	х	*
*	х	*	х	*	х	*	X
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
o	*	0	*	0	*	0	*
*	0	*	0	*	0	*	

crownPiece(BoardPosition posOfPlayer) – testCrownPiece_5_1_o

Inpu	t: pos	OfPla	ayer =	= (5, ′	1)				Outp	out: N	/A					
State	e: 8x8	B boa	rd						State	e: 8x8	3 boa	rd				
х	*	x	*	х	*	х	*		x	*	x	*	x	*	x	*
*	х	*	х	*	х	*	х		*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*		х	*	х	*	х	*	х	*
*		*		*		*			*		*		*		*	
	*		*		*		*			*		*		*		*
*	0	*	0	*	0	*	0		*	0	*	0	*	0	*	0
0	*	О	*	О	*	О	*		0	*	0	*	0	*	0	*
*	О	*	0	*	О	*			*	0	*	0	*	0	*	
	•	•	•	•	•	•	•	'		•	•	•	•	•	•	

 $crown Piece (Board Position\ pos Of Player) - test Crown Piece _1 _ Max Column _ x$

Input: posOfPlayer = (1, 7) Output: N/A State: 8x8 board State: 8x8 board Χ Χ Χ Χ Χ Χ Χ Χ Χ Χ Χ Χ Χ Χ Х Χ Χ Χ Χ

*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	О
o	*	0	*	0	*	0	*	О	*	0	*	0	*	0	*
*	0	*	О	*	0	*		*	0	*	0	*	0	*	
	-														

crownPiece(BoardPosition posOfPlayer) – testCrownPiece_MaxRow_1_o

Input: posOfPlayer = (7, 1)

State: 8x8 board

х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	0	*	О	*	О	*	0
o	*	0	*	О	*	0	*
*	0	*	o	*	0	*	

Output: N/A

State: 8x8 board

х	*	х	*	Х	*	х	*
*	х	*	х	*	х	*	Х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
О	*	0	*	0	*	0	*
*	O	*	0	*	0	*	

 $movePiece (BoardPosition\ startingPos,\ DirectionEnum\ dir)-testMovePiece_maxRow_maxCol_X$

Input	:							Outp	ut:						
start	ingP	os = ((6, 6)					mov	⁄ePie	ce =	(7,7)				
dir =	SE							State	e:						
State	e:										1	1	1		
Call	olace	Piece	e(star	tingP	os, 'x	<i>(</i> ')		х	*	х	*	х	*	х	*
			e(star)	*	х	*	x	*	x	*	х
oun _l	01400	7 7000) (Olar	ungi	00.0	, <u> </u>	,	Х	*	х	*	х	*	х	*
х	*	х	*	x	*	х	*	*		*		*		*	
*	х	*	x	*	х	*	x		*		*		*		*
х	*	Х	*	Х	*	Х	*	*	0	*	0	*	0	*	0
*		*		*		*		О	*	0	*	0	*		*
	*		*		*		*	*	0	*	0	*	0	*	X
*	0	*	0	*	0	*	0								
0	*	0	*	0	*	X	*								
*	0	*	0	*	0	*									
			•				•								

 $move Piece (Board Position\ starting Pos,\ Direction Enum\ dir)-test Move Piece_min Row_min Col_O$

															_
Inpu	t:							Outp	out:						
star	tingP	os =	(1, 1)					mo\	/ePie	ce =	(0,0)				
dir =	NW							State	e:						
State	e:							O	*	x	*	x	*	x	
Call	nlace	Piec	e/sta	rtinaE	Pos, '(つ ")		*		*	х	*	х	*]
							0	х	*	х	*	х	*	х	Ţ.
Call	place	Piec	e(sta	rtingF	Pos+N	VVV, ·)	*		*		*		*	T
	Ι.		Τ.	1	Ι.	T	Ι.		*		*		*		<u> </u>
	*	Х	*	Х	*	X	*	*	0	*	0	*	0	*	T,
*	0	*	X	*	Х	*	Х	0	*	0	*	0	*	0	t,
Х	*	Х	*	Х	*	Х	*	*	0	*	0	*	0	*	T
*		*		*		*								<u> </u>	L
	*		*		*		*								
*	0	*	0	*	0	*	0								
0	*	0	*	0	*	0	*								
*	0	*	0	*	0	*	0								

movePiece(BoardPosition startingPos, DirectionEnum dir) –testMovePiece_2_2_kingBackwards_O

Input	::							Outp	ut:						
starti	ngPo	os = (2	2,2)					mov	ePied	e = (3, 3)				
dir =	SE														
State	e:							State	e:						
Call _l	olace	Piece	e(star	tingP	os, '(O')									
Call _l	olace	Piece	e(star	tingP	os+S	SE, ''))								
х	*	х	*	х	*	х	*	Х	*	х	*	х	*	Х	*
*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	х
х	*	0	*	х	*	х	*	х	*		*	х	*	х	*
*		*		*		*		*		*	O	*		*	
	*		*		*		*		*		*		*		*
*	0	*	0	*	0	*	О	*	0	*	0	*	0	*	0
О	*	О	*	0	*	О	*	0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testJumpPiece_maxRow_maxCol_X

Inpu	rtingPos = (5, 5) = SE							Outp	ut:						
starti	ngPo	s = (5, 5)					jump	Piec	e = (7	7,7)				
dir =	SE														
State	e:							State	e:						
		_		_		_									
х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	,
*	х	*	х	*	х	*	х	*	х	*	х	*	х	*	,
х	*	х	*	х	*	х	*	х	*	х	*	х	*	х	,
*		*		*		*		*		*		*		*	
	*		*		*		*		*		*		*		,
*	0	*	0	*	X	*	О	*	0	*	0	*		*	0
0	*	0	*	o	*	0	*	0	*	0	*	0	*		,
*	О	*	0	*	0	*		*	0	*	0	*	0	*	>

 $jump Piece (Board Position\ starting Pos,\ Direction Enum\ dir)-test Jump Piece_min Row_min Col_O$

	Input: startingPos = (2, 2)								Output: jumpPiece(0,0)							
			(2, 2)	,					juiii	JF ICC	. c (0,0	,,				
air =	: NW															
State	State:							State	e :							
								.								
	*	х	*	х	*	х	*		O	*	х	*	х	*	х	*
*	х	*	x	*	х	*	х		*		*	x	*	x	*	x
х	*	O	*	х	*	х	*		х	*		*	х	*	х	*
*		*		*		*			*		*		*		*	
	*		*		*		*			*		*		*		*
*	0	*	0	*	0	*	0		*	0	*	0	*	0	*	О
0	*	0	*	0	*	0	*		0	*	0	*	0	*	0	*
*	0	*	О	*	0	*	О		*	0	*	0	*	0	*	О
<u> </u>		-	-	-	-	-								-		

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testJumpPiece_3_3_kingBackwardsO

Inpu	Input:								Output:							
star	startingPos = (1, 1)						jum	pPied	ce(3,3	3)						
dir =	dir = SE						State	e:								
State	e:															
									х	*	х	*	х	*	х	*
х	*	х	*	х	*	х	*		*		*	х	*	х	*	х
*	O	*	х	*	х	*	х		x	*		*	х	*	х	*
х	*	х	*	х	*	х	*		*		*	0	*		*	
*		*		*		*				*		*		*		*
	*		*		*		*		*	0	*	0	*	0	*	0
*	О	*	0	*	О	*	0		0	*	О	*	0	*	0	*
0	*	О	*	О	*	О	*		*	o	*	0	*	0	*	0
*	О	*	0	*	0	*	0									

 $playerLostPieces (int \ numPieces, \ char \ player, \ HashMap \ pieceCounts) - testPlayerLostPieces_1_x_1$

Input:	Output: N/A
numPieces = 1	pieceCounts.get('x') = 0
player = 'x'	
pieceCounts = {('x', 1)}	
State:	State:

pieceCounts = {(x	(, 1),	(0,	3)}

pieceCounts = $\{(x, 0), (o, 3)\}$

 $scan Surrounding Positions (Board Position\ starting Pos) - test Scan Surrounding Positions \underline{2_4_x}$

Input:

startingPos = (2, 4)

State:

х	*	Х	*	х	*	Х	*
*	х	*	х	*	х	*	х
х	*	х	*	×	*	х	*
*		*		*		*	
	*		*		*		*
*	О	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output:

 $scanSurroundingPositions = \{(NE: `x'), \\ (NW: `x'), (SE: ` `), (SW: ` `)\}$

scanSurroundingPositions(BoardPosition startingPos) – testScanSurroundingPositions_minRow_5_x

Input:

startingPos = (0, 5)

State:

х	*	х	*	×	*	х	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	О	*	О	*	0	*	0
О	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output:

scanSurroundingPositions = {(SE: 'x'),
(SW: 'x')}

scanSurroundingPositions(BoardPosition startingPos) – testScanSurroundingPositions_6_minColumn_x

In	nı	ıŧ
111	νι	Jι

startingPos = (6, 0)

State:

х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	О	*	О	*	О	*	О
0	*	0	*	О	*	0	*
*	0	*	0	*	0	*	o

Output:

scanSurroundingPositions = {(NE: 'o'),
(SE: 'o')}

scanSurroundingPositions(BoardPosition startingPos) – testScanSurroundingPositions_maxRow_3_o

Ir	าทเ	ıt.
ш	ıpı	Jι.

startingPos = (7, 3)

State:

х	*	х	*	x	*	х	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	О	*	0	*	О	*	0
o	*	0	*	0	*	0	*
*	0	*	O	*	0	*	0

Output:

scanSurroundingPositions = {(NE: 'o'),
(NW: 'o')}

scanSurroundingPositions(BoardPosition startingPos) – testScanSurroundingPositions_1_maxColumn_o

Input:

startingPos = (1, 7)

State:

х	*	х	*	x	*	х	*
*	х	*	x	*	х	*	×
х	*	х	*	x	*	х	*
*		*		*		*	
	*		*		*		*
*	О	*	0	*	О	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	О	*	0

Output:

scanSurroundingPositions = {(NW: 'x'),
(SW: 'x')}

getDirection(DirectionEnum dir) - testGetDirection_NE

Input:	Output:
dir = NE	new BoardPosition (-1, 1)
State:	State:

What tests did each team member write? Just tell me the names of the functions (unless for some reason multiple team members wrote functions for the same method. In that case, tell me which tests specifically by giving me the test names)

James Schvaneveldt	CheckerBoard, placePiece, scanSurroundingPositions
Isaac Beres	✓ whatsAtPos☐ movePiece (sorta)✓ getRowNum✓ getColNum
Nicolas Lozano	jumpPiece, checkPlayerWin, getViableDirections, addViableDirections
Liam Cassidy	crownPiece, playerLostPieces, getPieceCounts, getDirection