CheckerBoard(void) – [Test_Constructor]

Input: N/A State: N/A	Output: N/A State: pieceCount = x = 12, o = 12 viableDirections = x = SW, SE; o = NW, NE Board =

Board (Output):

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MaxRow_MinCol_BlackTile

Input: (7,0)

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: '*'

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MinRow_MinCol_WhiteTile

Input: (0,0)

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: 'x'

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MaxRow_MaxCol_BlackTile

Output: '*'

State:

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Input: (7, 7)

State:

X	*	X	*	X	*	X	*(highlighted)
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MinRow_MaxCol_WhiteTile

Input: (0, 7)

State:

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o(highlighted)

Output: 'o'

State:

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MidRow_MidCol_WhiteTile

Input: (3, 3)

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: ' '

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

placePiece(BoardPosition pos, char player) – [test_placePiece_pos07_x]

Input: [0, 7] x

State:

Board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: n/a

State:

Board =

x	*	x	*	x	*	x	x
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

placePiece(BoardPosition pos, char player) – [test_placePiece_pos00_o]

Input: [0,0] o

State:

Board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: n/a

State:

Board =

o	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

placePiece(BoardPosition pos, char player) – [test_placePiece_pos70_x]

Input: [7,0] x

State:

Board =

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: n/a

State:

Board =

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
X	o	*	o	*	o	*	o

placePiece(BoardPosition pos, char player) – [test_placePiece_pos77_x]

Input: [7, 7] x

State:

Board =

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: n/a

State:

Board =

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	O	*	o	*	o	*
*	o	*	o	*	o	*	X

placePiece(BoardPosition pos, char player) – [test_placePiece_pos44_o]

<table>
<tr><td>Input: [4, 4] o</td><td>Output: n/a</td></tr>
</table>

Input: [4, 4] o

State:

Board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: n/a

State:

Board =

x	*	X	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	X	*	x	*	x	*
*		*		*		*	
	*		*	o	*		*
*	o	*	o	*	o	*	o
o	*	O	*	o	*	o	*
*	o	*	o	*	o	*	o

getPieceCounts(void) – [Test_getPieceCounts_12]

Input: N/A

State:
pieceCount = x = 12, o = 12
viableDirections = x = SW, SE; o = NW, NE
board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: 12, 12

State:
pieceCount = x = 12, o = 12
viableDirections = x = SW, SE; o = NW, NE
board =

x	*	X	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	X	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	O	*	o	*	o	*
*	o	*	o	*	o	*	o

getViableDirections(void) – [Test_getViableDirections_start]

X	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
X	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
O	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Input: N/a

State:
pieceCount = x = 12, o = 12
viableDirections = x = SW, SE; o = NW, NE
board =
(above left table)

Output: x = SW, SE; o = NW, NE

State:
pieceCount = x = 12, o = 12
viableDirections = x = SW, SE; o = NW, NE
board =

x	*	X	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	X	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	O	*	o	*	o	*
*	o	*	o	*	o	*	o

checkPlayerWin(Character player) – [test_checkPlayerWin_playerXWin]

Input:
Character player = "x"

State:
Board contains only "x" pieces/ all "o" pieces have been captured.

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

Output:
Returns true

State:
No change.

checkPlayerWin(Character player) – [test_checkPlayerWin_playerXLosing]

Input: Character player = "x" State: Board contains "o" pieces and a singular "x" piece							Output: Returns false State: No change.

	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

crownPiece(BoardPosition posOfPlayer) – [test_crownPiece_playerCrown]

Input: BoardPosition(7, 1) State: Piece is lowercase ("x") at position (7, 1)							Output: N/A State: Position should be replaced with an uppercase character (so "x" becomes "X" to reflect the crowned piece).

Left board:

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Right board:

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	O	*	o	*	o	*	o

crownPiece(BoardPosition posOfPlayer) – [test_crownPiece_noCrown]

Input:
BoardPosition(3, 3)

State:
Piece is lowercase at position (3, 3)

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*	(yellow)	*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output:
N/A

State:
No change.

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

crownPiece(BoardPosition posOfPlayer) – [test_crownPiece_edge]

Input:
BoardPosition (0, 0)

State:
Piece is lowercase at position (0, 0)

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output:
N/A

State:
Position should be replaced with an uppercase character (so "o" becomes "O" to reflect the crowned piece).

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

crownPiece(BoardPosition posOfPlayer) – [test_crownPiece_crownTwice]

<table>
<tr><td>Input:
BoardPosition (0, 0)

State:
Piece is uppercase at position (0, 0)</td><td>Output:
N/A

State:
No change.</td></tr>
</table>

Input:
BoardPosition (0, 0)

State:
Piece is uppercase at position (0, 0)

X	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*		*		*		*	
	*		*		*		*
*	O	*	O	*	O	*	O
O	*	O	*	O	*	O	*
*	O	*	O	*	O	*	O

Output:
N/A

State:
No change.

movePiece(BoardPosition startingPos, DirectionEnum dir) – [test_movePiece_pos00]

Input: [0, 0] SE

State:
Board =

X	*	X	*	X	*	X	*
*		*	X	*	X	*	X
X	*	X	*	X	*	X	*
*	X	*		*		*	
	*		*		*	O	*
*	O	*	O	*	O	*	O
O	*	O	*	O	*		*
*	O	*	O	*	O	*	O

Output: [1,1]

State:
Board =

	*	X	*	X	*	X	*
*	X	*	X	*	X	*	X
X	*	X	*	X	*	X	*
*	X	*		*		*	
	*		*		*		*
*	O	*	O	*	O	*	O
O	*	O	*	O	*	O	*
*	O	*	O	*	O	*	O

movePiece(BoardPosition startingPos, DirectionEnum dir) – [test_movePiece_pos77]

Input: [7, 7] NW

State:

Board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*		*
*	o	*	o	*	o	*	o

Output:[6,6]

State:

Board =

x	*	X	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	X	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	

movePiece(BoardPosition startingPos, DirectionEnum dir) – [test_movePiece__pos53]

Input: [5, 3] NE

State:

Board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: [4,4]

State:

Board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*	o	*		*
*	o	*		*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – [Test_jumpPieceFrom_13SW]

Input:[1,3] SW

State:
Board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: [2,2]

State:
Board =

x	*	x	*	x	*	x	*
*	x	*		*	x	*	x
x	*		*	x	*	x	*
*	x	*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	O	*	o	*	o	*
*	o	*	o	*	o	*	o

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – [Test_jumpPieceFrom_60NE]

Input:[6,0] NE

State:
Board =

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: [4,2]

State:
Board =

x	*	X	*	x	*	x	*
*	x	*	x	*	x	*	X
x	*	X	*	x	*	x	*
*		*		*		*	
	*	O	*		*		*
*		*	o	*	o	*	O
	*	O	*	o	*	o	*
*	o	*	o	*	o	*	O

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – [Test_jumpPieceFrom_13SW]

Input: [1,3] SE

State:
Board =

x	*	x	*	x	*	x	*
*	x	*	X	*	x	*	x
x	*	x	*	x	*	x	*
*		*			*		*
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: [3,5]

State:
Board =

x	*	x	*	x	*	x	*
*	x	*		*	x	*	X
x	*	x	*		*	x	*
*		*		*	X	*	
	*		*		*		*
*	o	*	o	*	o	*	O
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	O

scanSurroundingPositions(BoardPosition startingPos) –
test_scanSurroundingPositions_MidRow_MidCol_WhiteTile

Input: (3,3)

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output:
DirectionEnum.SE = ' '
DirectionEnum.NE = 'x'
DirectionEnum.SW = ' '
DirectionEnum.NW = 'x'

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	O
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	O

scanSurroundingPositions(BoardPosition startingPos) –
test_scanSurroundingPositions_minRow_MaxCol_BlackTile

Input: (0, 7)

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output:
DirectionEnum.SE = N/A
DirectionEnum.NE = N/A
DirectionEnum.SW = '*'
DirectionEnum.NW = N/A

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	X
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

scanSurroundingPositions(BoardPosition startingPos) –
test_scanSurroundingPositions_MaxRow_SecondCol_WhiteTile

Input: (7, 1)

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output:
DirectionEnum.SE = N/A
DirectionEnum.NE = 'o'
DirectionEnum.SW = N/A
DirectionEnum.NW = 'o'

State:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	**X**
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	**O**
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

getDirection(DirectionEnum dir) – [test_getDirection_southeastSE]

Input:	Output:
DirectionEnum.SE	Returns BoardPosition(4, 4)
	(Position has moved 1 row down, 1 column right)
State:	
Piece is at position (3, 3)	State:
	No change.

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)

Elise James	whatsAtPos(BoardPosition) scanSurroundingPositions(BoardPosition)
Jada Hall	placePiece(BoardPosition, char) movePiece(BoardPosition, DirectionEnum)
Molly McKenzie	CheckerBoard(int) getPieceCounts(void) getViableDirections(void) jumpPiece(BoardPosition, Direction Enum)
Sarah Mendoza	getRowNum(void) getColNum(void) checkPlayerWin(Character) crownPiece(BoardPosition) getDirection(DirectionEnum)