CheckerBoard(int aDimension) – testCheckerBoardMinimumSize

Output:
State:
x * x * x * x *
* x * x * x * x
x * x * x * x *
* * * *
* 0 * 0 * 0
0 * 0 * 0 *
* 0 * 0 * 0

 $Checker Board (int\ a Dimension) - test Checker Board Maximum Size_16x16$

Input: aDimension = 16 State: N/A	Output: State: [State of the board is unchanged. Just a 16x16 game board instead of 8x8.]
	a Tox to game board instead of oxo.]

CheckerBoard(int aDimension) – testCheckerBoardInvalidSize

Input: aDimension = 10	Output:
State: N/A	State: [State of the board is unchanged. Just a 10x10 game board instead of 8x8.]

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MaxRow_MaxCol

Input: pos = (7, 7)	Output: 'o'
State:	State: [State of the board is unchanged.]
x * x * x * x *	
* x * x * x * x	
x * x * x * x *	
* * * *	
* * * *	
* 0 * 0 * 0	
0 * 0 * 0 * 0 *	
* 0 * 0 * 0 * 0	

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MaxRow_MinCol

Input: pos = (7, 0)	Output:
State: x * x * x * x * * x * x * x * x x * x * x * x * x * x * x * x * * *	State: [State of the board is unchanged.]

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MinRow_MaxCol

Input: pos = (0, 7)	Output:
State: x * x * x * x * * x * x * x * x x * x * x * x * x * x * x * x * * *	State: [State of the board is unchanged.]

$whats At Pos (Board Position\ pos) - test Whats At Pos_Min Row_Min Col$

Input:	Output:
pos = (0, 0)	'x '
State: x * x * x * x * * x * x * x * x x * x * x * x * x * x * x * x * * *	State: [State of the board is unchanged.]

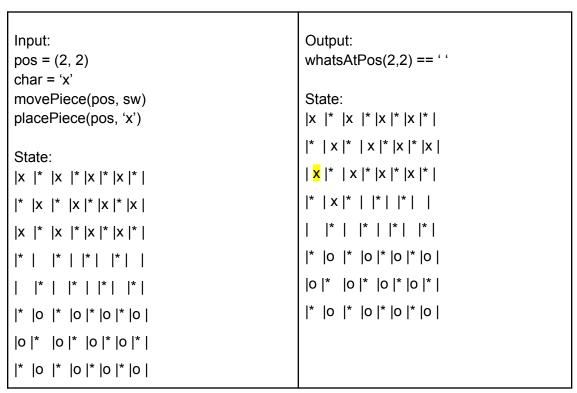
$whats At Pos (Board Position\ pos)-whats At Pos_Middle Of The Board$

Input: pos: (1, 6)	Output:
State: x * x * x * x * * x * x * x * x x * x * x * x * x * x * x * x * * *	State: [State of the board is unchanged.]

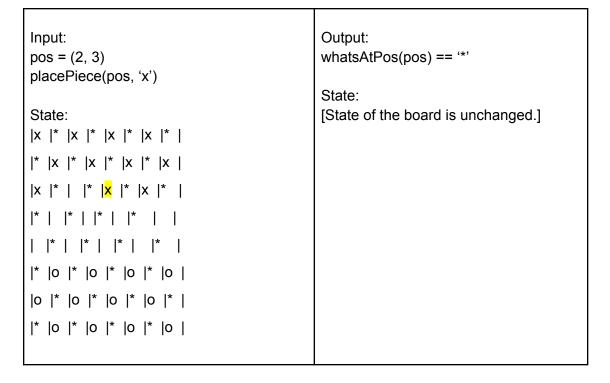
placePiece(BoardPosition pos, char player) – testPlacePieceInMiddle

Input: pos: (4, 0) char: 'x'	Output: whatsAtPos(pos) == 'x'
State: x * x * x * x * * x * x * x * x x * x * x * x x * x * x * x * * *	State: x * x * x * x * * x * x * x * x x * x * x * x * x * x * x * x * * *

placePiece(BoardPosition pos, char player) – testPlacePiece_OnPreviouslyOccupiedSpace



placePiece(BoardPosition pos, char player) – testOccupiedSpot



placePiece(BoardPosition pos, char player) - testPlacePieceMinColumn

Input: pos = (2, MinCol) char = 'o'	Output: void
State: * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *	State: * * * * * * * o * * * * * * * *
	* * *

placePiece(BoardPosition pos, char player) – testPlacePieceMinRow

Input: pos = (MinRow, MinCol) char = 'o'	Output: void
State:	State:
* * * *	O * * * *
* * * *	* * * *
* * * *	

placePiece(BoardPosition pos, char player) – testPlacePieceOnBlackSquare

Input: pos = (2, 1) char = 'x'	Output: void
	State:
State:	[State of the Board is unchanged.]
x * x * x * x *	
* x * x * x * x	
x <mark>*</mark> x * x * x *	
* * *	
* * * *	
* 0 * 0 * 0	
0 * 0 * 0 * 0 *	
* 0 * 0 * 0 * 0	

getPieceCounts(void) - testGetInitialPieceCounts

Input: N/A	Output: getPieceCounts(playerOne) == 12;
State:	getPieceCounts(playerTwo) == 12;
* x * x * x * x	State: [State of the board is unchanged.]
	[etate of the sound to unfoldingout]
* * * *	
* * * *	
* 0 * 0 * 0 * 0	
0 * 0 * 0 * 0 *	
* 0 * 0 * 0	

get Viable Directions (void) - test Get Initial Viable Directions

Input: N/A	Output: getViableDirections(playerTwo) == [NE,NW]
State:	
x * x * x * x *	State:
* x * x * x * x	[State of the board is unchanged.]
x * x * x * x *	
* * * *	
* 0 * 0 * 0 * 0	
* 0 * 0 * 0	

$add Viable Directions (char \ player, \ Direction Enum \ dir) - test Initial Direction Addition$

Input: player = 'x' dir = DirectionEnum.SE	Output: getViableDirections(playerOne) == [SE]
State: The viableDirections map is empty for player 'x' showing no directions have been assigned yet	State: The viableDirections map for player'x' now contains SE, showing that SE has been added as a viableDirection for player 'x'

getRowNum(void) - testGetRowNumForBoard

Input: N/A State:	Output: getRowNum = '8'
x * x * x * x * * x * x * x * x x * x * x * x * x * *	State: [State of the board is unchanged.]
* 0 * 0 * 0 * 0 0 * 0 * 0 * 0 * * 0 * 0 * 0	

getColNum(void) - testColRowNumForBoard

Input: N/A State:	Output: getColNum = '8'
	State:
* x * x * x * x	[State of the board is unchanged.]
x * x * x * x *	
* * *	
* * * *	
* 0 * 0 * 0	
0 * 0 * 0 * 0 *	
* 0 * 0 * 0 * 0	

checkPlayerWin(Character player) – testCheckPlayerOneWins

Input:	Output:
player = 'x'	True
State: * * * * * * *	State: [State of the board is unchanged.]

$check Player Win (Character\ player) - test Check No One Wins$

Input:	Output:
player = 'x'	False
State: * * * * * * *	State: [State of the board is unchanged.]

$crown Piece (Board Position\ pos Of Player) - test Crown Large Board$

Input:	Output:
posOfPlayer = (15,14)	void
State: A player one regular piece at position (15,14) on a 16x16 board.	State: Player one's 'x' becomes crowned, and switches to 'X', nothing else on the board changes.

$crown Piece (Board Position\ pos Of Player) - test Crown Player Two$

Input:	Output:
posOfPlayer = (1,1)	void
State: * x * x * x * * o * x * x * x x * * x * x * * *	State: O * x * x * x * * * x * x * x x * *

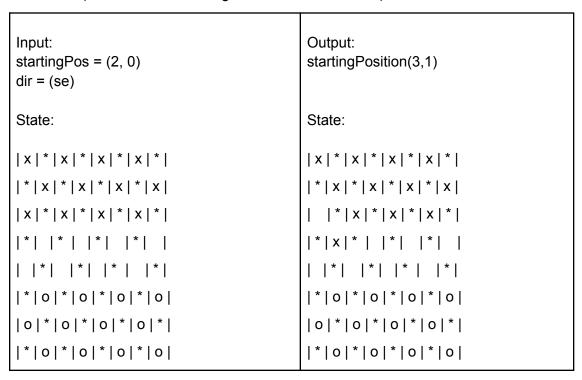
$crown Piece (Board Position\ pos Of Player) - test Crown Already Crowned Piece$

Input: posOfPlayer = (6,2)	Output: void
State:	State:
x * x * x * x *	x * x * x * x *
* x * x * x * x	* x * x * x * x
* * x * x *	* * x * x *
* * * *	* * * *
* * * *	* * * *
* * 0 * 0	* * 0 * 0 * 0
o * X * o * o *	0 * * 0 * 0 *
* 0 * * 0 * 0	* 0 * X * 0 * 0

crownPiece(BoardPosition posOfPlayer) – testCrownPieceMinRowMinCol

Input: posOfPlayer = (MinRow, MinCol)	Output: void
State: * x * x * x *	State: O * x * x * x *
* 0 * x * x * x	* * x * x * x
x * * x * x *	x * * x * x *
* * *	* * *
* * *	
* * 0 * 0 * 0	* * 0 * 0 * 0
0 * 0 * 0 *	0 * 0 * 0 *
* 0 * 0 * 0	* 0 * 0 * 0

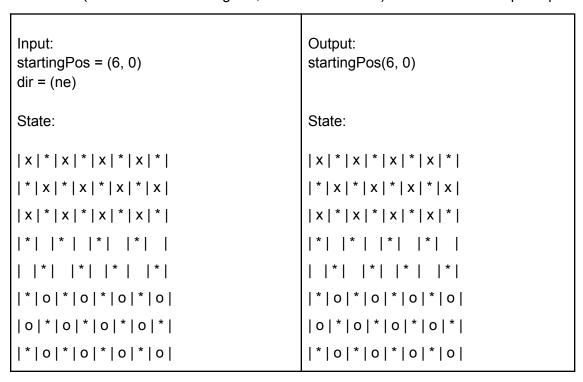
movePiece(BoardPosition startingPos, DirectionEnum dir) – testMoveToValidPos



movePiece(BoardPosition startingPos, DirectionEnum dir) – testMoveToOutOfBounds

Input: startingPos = (2, 0) dir = (sw)	Output: startingPos(2, 0)
State:	State:
	x * x * x * x *
* x * x * x * x	* x * x * x * x
x * x * x * x *	x * x * x * x *
* * * *	* * * *
* * * *	* * *
* 0 * 0 * 0	* 0 * 0 * 0
0 * 0 * 0 *	0 * 0 * 0 *
* 0 * 0 * 0	* 0 * 0 * 0

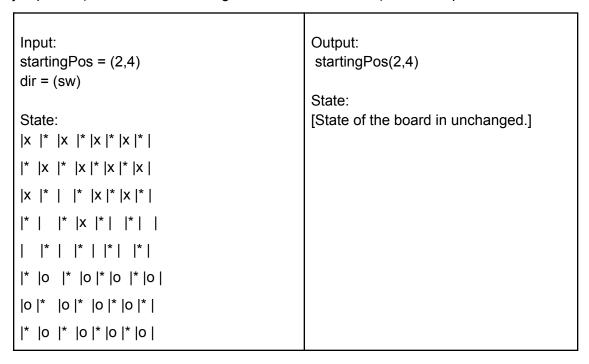
movePiece(BoardPosition startingPos, DirectionEnum dir) - testMoveToOccupiedSpace



jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testValidJump

Input: startingPos = (3, 3) dir = DirectionEnum.SW	Output: BoardPosition (4, 4)
State: x * x * x * x * * x * x * x * x x * *	State: x * x * x * x * * x * x * x * x x * *

jumpPiece(BoardPosition startingPos, DirectionEnum dir) - testJumpOwnPiece



jumpPiece(BoardPosition startingPos, DirectionEnum dir) - testJumpOffBoard

Input: startingPos = (7, 7) dir = DirectionEnum.SE	Output: startingPos(7, 7)
State: * * * * * * *	State: [State of the board is unchanged.]

playerLostPieces(int numPieces, char player, HashMap pieceCounts) – testPlayerOneLosesPieces

Input:
numPieces = 3
char = 'x'
pieceCounts: {'x': 0, 'o': 0}

State:
Player 'x' has all pieces

Output:
void

State:
pieceCounts: {'x': 3, 'o': 0}

 $scan Surrounding Positions (Board Position\ starting Pos)-test Scan Surrounding Positions At MaxRow$

```
Input:
                                                    Output:
startingPos = BoardPosition(7,3)
                                                    HashMap
                                                     NE = 'o'
State:
                                                     NW = 'o'
                                                     SE = EMPTY_POS
|x |* |x |* |x |* |x |* |
|* |x |* |x |* |x |* |x |
                                                     SW = EMPTY_POS
                                                    }
|x |* |x |* |x |* |x |* | |
| |* | |* | |* | |* | |
| | |* | |* | |* | |* |
|* |0 |* |0 |* |0 |* |0 |
                                                    State: [State of the board is unchanged]
|0 |* |0 |* |0 |* |0 |* |
|* |0 |* |<mark>0 |</mark>* |0 |* |0 |
```

scanSurroundingPositions(BoardPosition startingPos) – testScanSurroundingPositionsFromCenter

```
Output:
Input:
startingPos: BoardPosition(4,4)
                                               HashMap
State:
                                                NE = EMPTY POS
                                                NW = EMPTY_POS
                                                SE = 'o'
                                                SW = 'o'
|x |* |x |* |x |* |x |* |
|* |x |* |x |* |x |* |x |
                                               }
|x |* |x |* |x |* |x |* |
| |* | |* | |* | |* | |
State:[State of the board is unchanged]
|* |0 |* |0 |* |0 |* |0 |
|0 |* |0 |* |0 |* |0 |* |
|* |0 |* |0 |* |0 |* |0 |
```

scanSurroundingPositions(BoardPosition startingPos) – testScanFromNewlyCrownedKingPositionAtMinRow

```
Input:
                                                  Output:
startingPos: BoardPosition(0,4)
                                                  HashMap
State:
                                                   NE = EMPTY POS
An "O" at (0,4) indicates a Player Two
                                                   NW = EMPTY_POS
piece that has just been crowned as King.
                                                   SE = 'x'
                                                   SW = EMPTY_POS
|x |* |x |* |<mark>O</mark> |* |x |* |
                                                  }
|* |x |* | |* |x |* |x |
|x |* | |* |x |* |x |* |
                                                  State: [State of the Board is unchanged]
|* | |* |0 |* | |* | |
| |* |0 |* |0 |* | |* |
|* |0 |* | |* |0 |* |0 |
|0 |* | |* |0 |* |0 |* |
|* |0 |* |0 |* |0 |* | |
```

scanSurroundingPositions(BoardPosition startingPos) – testScanFromMaxCol

```
Input:
                                                     Output:
startingPos = BoardPosition(3,7)
                                                     HashMap
State:
                                                      NE = EMPTY_POS
                                                      NW = 'x'
                                                      SE = EMPTY POS
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
                                                      SW = 'o'
|0 |x |* |x |* |x |* |x |* |
                                                    }
|1 |* |x |* | |* |x |* |x |
|2 |x |* |0 |* | 0 |* |x |* |
|3 |* |x |* |o |* |x |* |<mark>o</mark> |
                                                     State: [State of the Board is
                                                     unchanged]
|4 |x |* |0 |* | |* |0 |* |
|5 |* | |* | |* | |* |
|6 |o |* | |* |o |* | |* |
|7 |* |0 |* |0 |* |0 |* |0 |
```

scanSurroundingPositions(BoardPosition startingPos) – test ScanSurroundingPositions AtMinCol.

```
Input:
                                      Output:
startingPos =
                                      HashMap
BoardPosition(2,0)
                                       NE = EMPTY_POS
                                       NW = EMPTY_POS
State:
                                       SE = 'x'
|<mark>x</mark> |* |x |* |x |* |x |* |
                                       SW = EMPTY_POS
|* |x |* | |* |x |* |x |
|x |* |0 |* | 0 |* |x |* |
|* |x |* |o |* |x |* |o |
|x |* |0 |* | |* |0 |* |
                                      State: [State of the Board is
|* | |* | |* | | |* | |
                                      unchanged]
0 |* | |* |0 |* | |* |
|* |0 |* |0 |* |0 |* |0 |
```

getDirection(DirectionEnum dir) - testGetAllDirections

Input: Output:

dir: DirectionEnum.NEBoardPosition(row: -1, col: 1)dir: DirectionEnum.NWBoardPosition(row: -1, col: -1)dir: DirectionEnum.SEBoardPosition(row: 1, col: 1)dir: DirectionEnum.SWBoardPosition(row: 1, col: -1)

State: N/A State: N/A

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)

[member 1]	checkerBoard(int aDimension) - all of them
Eli Boccolucci	whatsAtPos(BoardPosition pos) - testPieceAtPos
	whatsAtPos(BoardPosition pos) - testNoPieceAtPos
	whatsAtPos(BoardPosition pos) – testWhenAPieceJumpsAnother
	placePiece(BoardPosition pos, char player)-testPlacePieceInMiddle
	placePiece(BoardPosition pos, char player) - testOccupiedSpot
	placePiece(BoardPosition pos, char player)- testPlacePieceOnBlackSquare
	checkPlayerWin(Character player) – testCheckPlayerOnewins
	getViableDirections(void)
	getRowNum(void)
	jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testValidJump
	getDirection(DirectionEnum dir)
[member 2] Steven Spivack	placePiece(BoardPosition pos, char player) - testEdgeCaseRow
	placePiece(BoardPosition pos, char player) – testEdgeCaseColumn
	placePiece(BoardPosition pos, char player)
	testPlacePiece_OnPreviouslyOccupiedSpace
	crownPiece(BoardPosition posOfPlayer)

	movePiece(BoardPosition startingPos, DirectionEnum dir)
	whatsAtPos(BoardPosition pos) – testInvalidPosRow
	whatsAtPos(BoardPosition pos - AfterAPieceMovesFromPos
	whatsAtPos(BoardPosition pos - testWhatsAtPos_AfterJump
[member 3] Luke Miller	placePiece(BoardPosition pos, char player) – testJumpOwnPiece
	playerLostPieces(int numPieces, char player, HashMap pieceCounts) – testPlayerOneLosesPieces
	checkPlayerWin(Character player) – testCheckNoOneWins
	getPieceCounts(void)
[member 4] Jonathan Flander	- scanSurroundingPositions(BoardPos ition startingPos) - all of them
	- addViableDirections(char player, DirectionEnum dir)
	- placePiece(BoardPosition pos, char player)
	- jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testJumpOwnPiece
	- getColNum(void)