CheckerBoard(int aDimension) – testCheckerBoardMinimumSize

Input: aDimension = 8 State: N/A	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 |

CheckerBoard(int aDimension) – testCheckerBoardMaximumSize_16x16

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

CheckerBoard(int aDimension) – testCheckerBoardInvalidSize

Input: aDimension = 10 State: N/A	Output: 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) 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: 'o' State: [State of the board is unchanged.]

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MaxRow_MinCol

Input: pos = (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: [State of the board is unchanged.]

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MinRow_MaxCol

Input: pos = (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: '*' State: [State of the board is unchanged.]

whatsAtPos(BoardPosition pos) – testWhatsAtPos_MinRow_MinCol

Input: pos = (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: [State of the board is unchanged.]

whatsAtPos(BoardPosition pos) – whatsAtPos_MiddleOfTheBoard

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

placePiece(BoardPosition pos, char player) – testPlacePieceInMiddle

Input: pos: (4, 0) char: '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 |	Output: whatsAtPos(pos) == 'x' State: |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) – testPlacePiece_OnPreviouslyOccupiedSpace

Input: pos = (2, 2) char = ‘x’ movePiece(pos, sw) placePiece(pos, ‘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 |	Output: whatsAtPos(2,2) == ‘ ‘ State: |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) – testOccupiedSpot

Input: pos = (2, 3) placePiece(pos, ‘x’) State: |x |* |x |* |x |* |x |* | |* |x |* |x |* |x |* |x | |x |* | |* |x |* |x |* | |* | |* | |* | |* | | | |* | |* | |* | |* | |* |o |* |o |* |o |* |o | |o |* |o |* |o |* |o |* | |* |o |* |o |* |o |* |o |	Output: whatsAtPos(pos) == ‘*’ State: [State of the board is unchanged.]

placePiece(BoardPosition pos, char player) - testPlacePieceMinColumn

Input: pos = (2, MinCol) char = ‘o’ State: | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| |	Output: void State: | |*| |*| |*| |*| |*| |*| |*| |*| | |o|*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*||*| |*| |*| |

placePiece(BoardPosition pos, char player) – testPlacePieceMinRow

Input: pos = (MinRow, MinCol) char = ‘o’ State: | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| |	Output: void State: |o|*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| | | |*| |*| |*| |*| |*| |*| |*| |*| |

placePiece(BoardPosition pos, char player) – testPlacePieceOnBlackSquare

Input: pos = (2, 1) char = ‘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 |	Output: void State: [State of the Board is unchanged.]

getPieceCounts(void) – testGetInitialPieceCounts

Input: 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 |
	Output:
	getPieceCounts(playerOne) == 12;
	getPieceCounts(playerTwo) == 12;
	
	State:
	[State of the board is unchanged.]

getViableDirections(void) – testGetInitialViableDirections

Input: 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 |	Output: getViableDirections(playerTwo) == [NE,NW] State: [State of the board is unchanged.]

addViableDirections(char player, DirectionEnum dir) – testInitialDirectionAddition

Input: player = ‘x’ dir = DirectionEnum.SE State: The viableDirections map is empty for player ‘x’ showing no directions have been assigned yet	Output: getViableDirections(playerOne) == [SE] 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: |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: getRowNum = ‘8’ State: [State of the board is unchanged.]

getColNum(void) – testColRowNumForBoard

Input: 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 |	Output: getColNum = ‘8’ State: [State of the board is unchanged.]

checkPlayerWin(Character player) – testCheckPlayerOneWins

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

checkPlayerWin(Character player) – testCheckNoOneWins

Input: player = ‘x’ State: | |* | |* | |* | |* | |* | |* | |* | |* | |* |x | | |* | |* | |* | |* | |* | |* | |* | |* |x |* | |* | | | | |* | |* | |* |x |* | |* | |* | |* | |* | |* | |* | |* | | | | |* | |* | |* | |* | |* | | | |* | |* | |* | |* | |o |	Output: False State: [State of the board is unchanged.]

crownPiece(BoardPosition posOfPlayer) – testCrownLargeBoard

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

crownPiece(BoardPosition posOfPlayer) – testCrownPlayerTwo

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

crownPiece(BoardPosition posOfPlayer) – testCrownAlreadyCrownedPiece

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

crownPiece(BoardPosition posOfPlayer) – testCrownPieceMinRowMinCol

Input: posOfPlayer = (MinRow, MinCol) State: | |* |x |* |x |* |x |* | |* | o |* |x |* |x |* |x | |x |* | |* |x |* |x |* | |* | |* | |* | |* | | | |* | |* | |* | |* | |* | |* |o |* |o |* |o | |o |* |o |* |o |* |o |* | |* |o |* |o |* |o |* |o |	Output: void State: | O |* |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) – testMoveToValidPos

Input: startingPos = (2, 0) dir = (se) 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: startingPosition(3,1) 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 |

movePiece(BoardPosition startingPos, DirectionEnum dir) – testMoveToOutOfBounds

Input: startingPos = (2, 0) dir = (sw) 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: startingPos(2, 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 |

movePiece(BoardPosition startingPos, DirectionEnum dir) – testMoveToOccupiedSpace

Input: startingPos = (6, 0) dir = (ne) 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: startingPos(6, 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 |

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testValidJump

Input: startingPos = (3, 3) dir = DirectionEnum.SW 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: BoardPosition (4, 4) State: |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) – testJumpOwnPiece

Input: startingPos = (2,4) dir = (sw) 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: startingPos(2,4) State: [State of the board in unchanged.]

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testJumpOffBoard

Input: startingPos = (7, 7) dir = DirectionEnum.SE State: | |* | |* | |* | |* | |* | |* | |* | |* | | | | |* | |* | |* | |* | |* | |* | |* | |* | | | | |* | |* | |* | |* | |* | |* | |* | |* | | | | |* | |* | |* |x |* | |* | |* | |* | |* |o |	Output: startingPos(7, 7) 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}

scanSurroundingPositions(BoardPosition startingPos) – testScanSurroundingPositionsAtMaxRow

Input:
startingPos = BoardPosition(7,3)
 	
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:
	HashMap
	{
	NE = ‘o’
	NW = ‘o’
	SE = EMPTY_POS
	SW = EMPTY_POS
	}
	
	State: [State of the board is unchanged]

scanSurroundingPositions(BoardPosition startingPos) –
testScanSurroundingPositionsFromCenter

Input:
startingPos: BoardPosition(4,4)
 	
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:
	HashMap
	{
	NE = EMPTY_POS
	NW = EMPTY_POS
	SE = ‘o’
	SW = ‘o’
	}
	
	State:[State of the board is unchanged]

scanSurroundingPositions(BoardPosition startingPos) –
testScanFromNewlyCrownedKingPositionAtMinRow

Input:
startingPos: BoardPosition(0,4)
 	
State:
An “O” at (0,4) indicates a Player Two piece that has just been crowned as King.
 	
|x |* |x |* |O |* |x |* |
|* |x |* | |* |x |* |x |
|x |* | |* |x |* |x |* |
|* | |* |o |* | |* | |
| |* |o |* |o |* | |* |
|* |o |* | |* |o |* |o |
|o |* | |* |o |* |o |* |
|* |o |* |o |* |o |* | |
	Output:
	HashMap
	{
	NE = EMPTY_POS
	NW = EMPTY_POS
	SE = ‘x’
	SW = EMPTY_POS
	}
	
	State:[State of the Board is unchanged]

scanSurroundingPositions(BoardPosition startingPos) – testScanFromMaxCol

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

scanSurroundingPositions(BoardPosition startingPos) – test_ScanSurroundingPositions_AtMinCol.

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

getDirection(DirectionEnum dir) – testGetAllDirections

Input: dir: DirectionEnum.NE dir: DirectionEnum.NW dir: DirectionEnum.SE dir: DirectionEnum.SW State: N/A	Output: BoardPosition(row: -1, col: 1) BoardPosition(row: -1, col: -1) BoardPosition(row: 1, col: 1) BoardPosition(row: 1, col: -1) 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] Eli Boccolucci	checkerBoard(int aDimension) - all of them 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(BoardPosition startingPos) - all of them - addViableDirections(char player, DirectionEnum dir) - placePiece(BoardPosition pos, char player) - jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testJumpOwnPiece - getColNum(void)