$Checker Board (void) - test Check Board_Initialize\\$

Input: N/A

State: N/A

Output: [Board initialized]

State:

Otate							
Х	*	Х	*	Х	*	Х	*
*	Χ	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

 $whats At Pos (Board Position\ pos)-test Whats At Pos_Min Row_Min Col_Player_1$

Input: pos = (0, 0)

State	e:						
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: 'x'

State: [Board state is unchanged]

 $whats At Pos(Board Position\ pos)-test Whats At Pos_Max Row_Max Col$

Input: pos = (7, 7)

State:

State	•						
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: "o"

whatsAtPos(BoardPosition pos) – testWhatsAtPos_Empty_Space

Input: pos = (3, 1)

State	:						
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: ' '

State: [Board state is unchanged]

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

Input: pos = (0, 1)

State:

State	··						
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: '*'

State: [Board state is unchanged]

whatsAtPos(BoardPosition pos) - testWhatsAtPos_After_Turn_1_Player_1

Input: pos = (3, 3)

State: [Player 1 moved from (2.2) to (3.3)]

State	: [Pla	yer i	HIOVE	u IIOI	11 (2,2	<u>2) to (</u> ,	<u>ಎ,ಎ)]</u>
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*		*	Х	*	Х	*
*		*	Х	*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: 'x'

placePiece(BoardPosition pos, char player) – testPlacePiece_Empty_Tile _Player_1

Input: pos = (3, 1), player = 'x' Output: N/A State: Х Χ Χ Χ Χ Χ Χ Χ Χ Χ Х Х Х Х Χ Χ Х Х Χ Х Χ Х Χ Х Х 0 0 0 0 0 0 0 0 0 * * 0 0 0 0 0 0 0 0

placePiece(BoardPosition pos, char player) – testPlacePiece_Occupied_Tile_Empty

npu	ıt: pos	s = (5,	1), p	layer	=''			Out	out: N	N/A					
Stat	e:							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	*
*		*		*		*		*		*		*		*	1
	*		*		*		*		*		*		*		*
*	0	*	0	*	0	*	0	*		*	0	*	0	*	О
0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*
*	0	*	0	*	О	*	0	*	0	*	0	*	0	*	О

placePiece(BoardPosition pos, char player) – testPlacePiece_Empty_TIle_Player_2

Inpu	t: pos	s = (4,	6), p	layer	= '0'			Out	put: N	I/A					
State	e:							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	*
*		*		*		*		*		*		*		*	
	*		*		*		*		*		*		*	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

placePiece(BoardPosition pos, char player) – testPlacePiece_After_Turn_1_Player_2

Inpu	t: pos	; = (2,	4), pl	.ayer :	= '0'				Outp		I/A					
Х	*	Х	*	Х	*	Х	*		Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х		*	Х	*	Х	*	Х	*	Х
Х	*	Х	*		*	Х	*		Х	*	Х	*	0	*	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

placePiece(BoardPosition pos, char player) – testPlacePiece_After_Turn_2_Player_1

Inpu	t: pos	s = (5,	7), p	layer	= ' x'			(Outp	out: N	I/A					
State	e:							,	State	e:						
Х	*	Х	*	Х	*	Х	*		Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х	-	*	Х	*	Х	*	Х	*	Х
Х	*	Х	*		*	Х	*	-	Χ	*	Х	*		*	Х	*
*		*		*	Х	*		-	*		*		*	Х	*	
	*		*		*	0	*	-		*		*		*	0	*
*	0	*	0	*	0	*			*	0	*	0	*	0	*	Х
О	*	0	*	0	*	О	*		0	*	0	*	0	*	0	*
*	0	*	0	*	О	*	0		*	0	*	0	*	0	*	0

$getPieceCounts(void) - getPieceCounts_Initial_Board$

Input: N/A

State.

State	•						
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: {'x'=12, 'o'=12}

State: [Board state is unchanged]

get Viable Directions (void) - test Get Viable Directions

Input: N/A

State):						
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: $\{'x'=[SE, SW], 'X'=[SE, SW, NE, SE, SW], 'X'=[SE, SW, NE, SW], 'X'=[SE, SW, NE, SW], 'X'=[SE, SW, NE, SW], 'X'=[SE, SW, NE, SW], 'X'=[SE, SW], 'X'=[SE, SW, NE, SW], 'X'=[SE, SW, SW], 'X'=[SE, SW], 'X'=[SE$ NW], 'o'=[NE, NW], 'O'=[SE, SW, NE, NW]}

State: [Board state is unchanged]

checkPlayerWin(Character player) - testCheckPlayerWin_False_Player_1

Input: player = 'x'

State.

State	:						
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: false

$check Player Win (Character\ player) - test Check Player Win_True_Player_1$

Ir	nput	: N/A							Output: true
S	tate	:							State: [Board state is unchanged]
	Χ	*	Х	*	Х	*	Х	*	
	*	Х	*	Х	*	Х	*	Х	
	Χ	*	Х	*	Х	*	Х	*	
	*		*		*		*		
		*		*		*		*	
	*		*		*		*		
		*		*		*		*	
	*		*		*		*		

crownPiece(BoardPosition posOfPlayer) – testCrownPiece_Back_Row_Player_1

Inp	ut: po:	sOfPl	ayer =	= (7, 1)			Outp	out:						
Sta	te:							State	e:						
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	*
*	Х	*	0	*	0	*	0	*	Х	*	0	*	0	*	0

 $crown Piece (Board Position\ pos Of Player) - test Crown Piece _Middle_Board_Player_1$

Inpu	t: pos	sOfPl	ayer =	= (3, 3)			Outp	out: N	I/A					
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	*		*		*		*	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
							<u>.</u>								

 $crown Piece (Board Position\ pos Of Player) - test Crown Piece_Back_Row_Player_2$

Inpu	t: pos	OfPla	ayer =	(0, 2)			(Outp	ut: N	I/A					
State	e:							,	State	e:						
Х	*	0	*	Х	*	Х	*		Х	*	0	*	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

 $move Piece (Board Position\ starting Pos,\ Direction Enum\ dir)-test Movie Piece_Player_1_SE$

Inpu	t: staı	rtingF	Pos =	(2, 0),	dir=	SE		Outp	out: (3, 1)					
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	*
*		*		*		*		*	Х	*		*		*	
	*		*		*		*		*		*		*		*
*	0	*	0	*	0	*	0	*	О	*	0	*	0	*	0
0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	О

movePiece(BoardPosition startingPos, DirectionEnum dir) – testMovePiece _Player_2_NE

Input	t: star	tingP	os = (5, 5),	dir=	NE		Outp	out: (4	4, 6)					
State	e:							State	e:						
Х	*	Х	*	Х	*	Х	*	Х	*	Х	*	Х	*	Х	
*	Х	*	Х	*	Х	*	Х	*	Х	*	Х	*	Х	*	Ī
Х	*	Х	*	Х	*	Х	*	Х	*	Х	*	Х	*	Х	Ī
*		*		*		*		*		*		*		*	Ī
	*		*		*		*		*		*		*	0	Ī
*	0	*	0	*	0	*	0	*	0	*	0	*		*	Ī
0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	Ī
*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	Γ

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

Input	:: star	tingP	os = ((4, 4),	dir=	SW		Out	out: (5, 3)					
State	e:							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	*		*		*		*		*		*		*	
	*		*	0	*		*		*		*		*		*
*	0	*		*	0	*	0	*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*	0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0	*	0	*	0	*	0	*	0

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testJumpPiece_Player_1_Jump_Player_2

Input: startingPos = (3, 3), dir = SW Output: (5, 1) State: State: Х Χ Х Χ Х Χ Х Χ Χ Х Х Χ Х Х Х Х Χ Χ Χ Х Х 0 О 0 0 Χ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

jumpPiece(BoardPosition startingPos, DirectionEnum dir) testJumpPiece_Player_2_Jump_Player_1

Input: startingPos = (5, 3), dir = NW Output: (3, 1) State: State: Х Χ Χ Х Х Х Х Χ Χ Х Х Х Х Χ Χ Х Х Χ Х Х Χ * * 0 Х 0 0 0 0 0 0 0 О О 0 0 0 0 0 * 0 0 0 0 0 jumpPiece(BoardPosition startingPos, DirectionEnum dir) – testJumpPiece_Crowned_Piece_Jump_Backwards

Input: startingPos = (5, 5), dir = NE

State	:						
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*		*
*		*		*		*	
	*		*		*	0	*
*	0	*	0	*	Χ	*	
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: (3, 7)

State:

Otate							
Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*		*
*		*		*		*	Χ
	*		*		*		*
*	0	*	0	*		*	
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

 $scan Surrounding Positions (Board Position \, starting Pos) - test Scan Surrounding Positions _All_Empty$

Input: startingPos = (4, 2)

State:

Ι.	- 10.10	•						
	Χ	*	Х	*	Х	*	Х	*
	*	Х	*	Х	*	Х	*	Х
	Х	*	Х	*	Х	*	Х	*
	*		*		*		*	
		*	0	*	0	*		*
	*		*		*	0	*	0
	0	*	0	*	0	*	0	*
	*	0	*	0	*	0	*	0

Output: {SE='', SW='', NE='', NW=''}

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

Input: startingPos = (6, 2)

State:

•	State	•						
	Χ	*	Х	*	Х	*	Х	*
	*	Х	*	Х	*	Х	*	Х
	Х	*	Х	*	Х	*	Х	*
	*		*		*		*	
Ī		*		*		*		*
	*	0	*	0	*	0	*	0
	0	*	0	*	0	*	0	*
	*	0	*	0	*	0	*	0

Output: {SE='o', SW='o', NE='o', NW='o'}

State: [Board state is unchanged]

 $scan Surrounding Positions (Board Position \, starting Pos) - test Scan Surrounding Positions_Board_Edge$

Input: startingPos = (2, 0)

State:

Х	*	Х	*	Х	*	Х	*
*	Х	*	Х	*	Х	*	Х
Х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: {SE='', SW="Off The Board", NE='x', NW="Off The Board"}