Project 4 Testing Report

GameBoard(int rows, int columns, int winNum)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**  rows = 3  columns = 3  winNum = 3 | **Output:**  **State:**   |  |  |  |  | | --- | --- | --- | --- | |  | **0** | **1** | **2** | | **0** |  |  |  | | **1** |  |  |  | | **2** |  |  |  | | **Reason:**  This test case is unique and distinct because 3 is the minimum number of rows and columns a gameboard can have.  **Function Name:**  testGameBoard\_min\_rows\_and\_columns |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**  Rows = 100  Columns = 100  winNum = 3 | **Output:**  **State:**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **…** | **98** | **99** | | **0** |  |  |  |  |  | | **1** |  |  |  |  |  | | **…** |  |  |  |  |  | | **98** |  |  |  |  |  | | **99** |  |  |  |  |  | | **Reason:**  This test case is unique and distinct because 100 is the maximum number of rows and columns a gameboard can have.  **Function Name:**  testGameBoard\_max\_rows\_and\_columns |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**  rows = 3  columns = 100  winNum = 3 | **Output:**  **State:**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **…** | **98** | **99** | | **0** |  |  |  |  |  | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **Reason:**  This test case is unique and distinct because 3 is the minimum of rows and 100 is the maximum of columns. Additionally, this game board accounts for rectangular shapes of the board  **Function Name:**  testGameBoard\_min\_rows\_and\_max\_columns |

boolean checkSpace(BoardPosition pos)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  | A |  |  | A | | 1 |  |  |  |  |  |  | | 2 | X |  |  |  |  |  | | 3 | X | X |  |  |  |  | | 4 |  |  |  |  |  |  | | 5 |  |  |  |  | O |  | | 6 |  |  |  | O |  |  |   pos.getRow() = 7  pos.getColumn() = 6 | **Output:**  checkSpace = false  state of the board is unchanged | **Reason:**  This test case is unique and distinct because the row and column of the space we are checking is not within bounds. A unique path.  **Function Name:**  testCheckSpace\_bad\_row\_and\_column |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  | A |  |  | A | | 1 |  |  |  |  |  |  | | 2 | X |  |  |  |  |  | | 3 | X | X |  |  |  |  | | 4 |  |  |  |  |  |  | | 5 |  |  |  |  | O |  | | 6 |  |  |  | O |  |  |   pos.getRow() = 3  pos.getColumn() = 3 | **Output:**  checkSpace = true  state of the board is unchanged | **Reason:**  This test case is unique and distinct because the space we are checking is not occupied. A unique path.  **Function Name:**  testCheckSpace\_empty |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  | A |  |  | A | | 1 |  |  |  |  |  |  | | 2 | X |  |  |  |  |  | | 3 | X | X |  |  |  |  | | 4 |  |  |  |  |  |  | | 5 |  |  |  |  | O |  | | 6 |  |  |  | O |  |  |   pos.getRow() = 0  pos.getColumn() = 2 | **Output:**  checkSpace = false  state of board is unchanged | **Reason:**  This test case is distinct and unique because it checks a space that is already occupied. A unique path.  **Function Name:**  testCheckSpace\_occupied |

boolean checkHorizontalWin(BoardPosition lastPos, char player)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  |  |  |  |  | | 1 |  |  |  |  | A |  | | 2 |  |  |  |  | A |  | | 3 | O |  |  |  | A |  | | 4 | O | O | **X** | X | X | X | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow() = 4  lastPos.getColumn() = 2  player = ‘X’  numToWin = 4 | **Output:**  checkHorizontalWin = true  state of board is unchanged | **Reason:**  This test case is distinct and unique because it counts the Xs to the right.  **Function Name:**  testCheckHorizontalWin\_right |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  |  |  |  |  | | 1 |  |  |  |  | A |  | | 2 |  |  |  |  | A |  | | 3 | O |  |  |  | A |  | | 4 | O | O | X | X | X | **X** | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow() = 4  lastPos.getColumn() = 5  player = ‘X’  numToWin = 4 | **Output:**  checkHorizontalWin = true  state of board is unchanged | **Reason:**  This test case is distinct and unique because it counts the Xs to the left.  **Function Name:**  testCheckHorizontalWin\_left |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  |  |  |  |  | | 1 |  |  |  |  | A |  | | 2 |  |  |  |  | A |  | | 3 | O |  |  |  | A |  | | 4 | O | O | X | **X** | X | X | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow() = 4  lastPos.getColumn() = 3  player = ‘X’  numToWin = 4 | **Output:**  checkHorizontalWin = true  state of board is unchanged | **Reason:**  This test case is distinct and unique because it counts the Xs to the left and right.  **Function Name:**  testCheckHorizontalWin\_middle |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  |  |  |  |  | | 1 |  |  |  |  | A |  | | 2 | O |  |  |  | A |  | | 3 | O | O |  |  | A |  | | 4 | O | X | **X** | X | X | X | | 5 | A |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow() = 4  lastPos.getColumn() = 2  player = ‘X’  numToWin = 4 | **Output:**  checkHorizontalWin = true  state of board is unchanged | **Reason:**  This test case is distinct and unique because it counts the Xs to the left and right win pieces aligned are greater than the number to win.  **Function Name:**  testCheckHorizontalWin\_super\_win |

boolean checkVerticalWin(BoardPosition lastPos, char player)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  |  |  |  |  | | 1 |  |  |  |  |  |  | | 2 |  |  |  |  |  |  | | 3 | **O** |  |  | A |  |  | | 4 | O |  |  | A |  |  | | 5 | O | X | X | X |  |  | | 6 | O |  |  | A |  |  |   lastPos.getRow = 3  lastPost.getcolumn = 0  player = ‘O’  numToWin = 4 | **Output:**  checkVerticalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts the same aligned pieces downwards.  **Function Name:**  testCheckVerticalWin\_down |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  |  |  |  |  | | 1 |  |  |  |  |  |  | | 2 |  |  |  |  |  |  | | 3 | O |  |  | A |  |  | | 4 | O |  |  | A |  |  | | 5 | O | X | X | X |  |  | | 6 | **O** |  |  | A |  |  |   lastPos.getRow = 6  lastPost.getcolumn = 0  player = ‘O’  numToWin = 4 | **Output:**  checkVerticalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts the same aligned pieces upwards.  **Function Name:**  testCheckVerticalWin\_up |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  |  |  |  |  | | 1 |  |  |  |  |  |  | | 2 |  |  |  |  |  |  | | 3 | O |  |  | A |  |  | | 4 | O |  |  | A |  |  | | 5 | **O** | X | X | X |  |  | | 6 | O |  |  | A |  |  |   lastPos.getRow = 5  lastPost.getcolumn = 0  player = ‘O’  numToWin = 4 | **Output:**  checkVerticalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts the same aligned pieces upwards and downwards.  **Function Name:**  testCheckVerticalWin\_middle |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 |  |  |  |  |  |  | | 1 |  |  |  |  |  |  | | 2 | O |  |  |  |  |  | | 3 | O |  |  | A |  |  | | 4 | **O** |  |  | A |  |  | | 5 | O | X | X | X |  |  | | 6 | O |  |  | A |  |  |   lastPos.getRow = 4  lastPost.getcolumn = 0  player = ‘O’  numToWin = 4 | **Output:**  checkVerticalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts alike aligned pieces downwards and upwards, while accounting for a count that exceeds the number to win.  **Function Name:**  testCheckVerticalWin\_super\_win |

boolean checkDiagonalWin(BoardPosition lastPos, char player)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | O | O |  |  |  |  | | 1 | O |  |  |  | A |  | | 2 | X | X | X | A |  |  | | 3 |  |  | A |  |  |  | | 4 |  | **A** |  |  |  |  | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow = 4  lastPost.getcolumn = 1  player = A  numToWin = 4 | **Output:**  checkDiagonalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts alike pieces going to the upper right diagonally.  **Function Name:**  testCheckDiagonalWin\_up\_right |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | O | O |  |  |  |  | | 1 | O |  |  |  | **A** |  | | 2 | X | X | X | A |  |  | | 3 |  |  | A |  |  |  | | 4 |  | A |  |  |  |  | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow = 1  lastPost.getcolumn = 4  player = A  numToWin = 4 | **Output:**  checkDiagonalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts alike pieces going to the lower left diagonally.  **Function Name:**  testCheckDiagonalWin\_low\_left |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | O | O |  |  |  |  | | 1 | O |  |  |  | A |  | | 2 | X | X | X | **A** |  |  | | 3 |  |  | A |  |  |  | | 4 |  | A |  |  |  |  | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow = 2  lastPost.getcolumn = 3  player = A  numToWin = 4 | **Output:**  checkDiagonalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts the alike pieces going upper right and lower left.  **Function Name:**  testCheckDiagonalWin\_up\_right\_and\_low\_left |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | O | O |  |  |  |  | | 1 | O |  | **A** |  |  |  | | 2 | X | X | X | A |  |  | | 3 |  |  |  |  | A |  | | 4 |  |  |  |  |  | A | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow = 1  lastPost.getcolumn = 2  player = A  numToWin = 4 | **Output:**  checkDiagonalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts alike pieces going towards the lower right.  **Function Name:**  testCheckDiagonalWin\_low\_right |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | O | O |  |  |  |  | | 1 | O |  | A |  |  |  | | 2 | X | X | X | A |  |  | | 3 |  |  |  |  | A |  | | 4 |  |  |  |  |  | **A** | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow = 4  lastPost.getcolumn = 5  player = A  numToWin = 4 | **Output:**  checkDiagonalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts pieces going towards the upper left.  **Function Name:**  testCheckDiagonalWin\_up\_left |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | O | O |  |  |  |  | | 1 | O |  | A |  |  |  | | 2 | X | X | X | **A** |  |  | | 3 |  |  |  |  | A |  | | 4 |  |  |  |  |  | A | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow = 2  lastPost.getcolumn = 3  player = A  numToWin = 4 | **Output:**  checkDiagonalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it counts pieces going towards the upper left and lower right.  **Function Name:**  testCheckDiagonalWin\_up\_left\_and\_low\_right |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | O | O |  |  |  | A | | 1 | O |  |  |  | A |  | | 2 | X | X | X | **A** |  |  | | 3 |  |  | A |  |  |  | | 4 |  | A |  |  |  |  | | 5 |  |  |  |  |  |  | | 6 |  |  |  |  |  |  |   lastPos.getRow = 2  lastPost.getcolumn = 3  player = A  numToWin = 4 | **Output:**  checkDiagonalWin = true  state of board is unchanged | **Reason:**  This test case is unique and distinct because it accounts for numbers of aligned pieces that exceed the number to win.  **Function Name:**  testCheckDiagonalWin\_superWin |

boolean checkForDraw()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  | | --- | --- | --- | --- | |  | **0** | **1** | **2** | | **0** | X | X | X | | **1** | X | X | X | | **2** | X | X | X | | **Output:**  checkForDraw = True  state of board is unchanged | **Reason:**  This test case is unique and distinct because it accounts for draws that may occur in minimally small boards.  **Function Name:**  testCheckForDraw\_small |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **…** | **98** | **99** | | **0** | X | X | X | X | X | | **1** | X | X | X | X | X | | **…** | X | X | X | X | X | | **98** | X | X | X | X | X | | **99** | X | X | X | X | X | | **Output:**  checkForDraw = True  state of board is unchanged | **Reason:**  This test case is unique and distinct because it accounts for draws that may occur in maximally large boards.  **Function Name:**  testCheckForDraw\_large |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  | | --- | --- | --- | --- | |  | **0** | **1** | **2** | | **0** | X | X | X | | **1** | X |  | X | | **2** | X | X | X | | **Output:**  checkForDraw = False  state of board is unchanged | **Reason:**  This test case is unique and distinct because it tests that all spaces are counted and not just the outside rows and columns.  **Function Name:**  testCheckForDraw\_hole |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  | | --- | --- | --- | --- | |  | **0** | **1** | **2** | | **0** |  |  |  | | **1** |  |  |  | | **2** |  |  |  | | **Output:**  checkForDraw = False  state of board is unchanged | **Reason:**  This test case is unique and distinct because it tests that empty pieces alike are not mistaken for a draw. Actual player pieces are counted and not just cells inside the grid.  **Function Name:**  testCheckForDraw\_empty |

char whatsAtPos(BoardPosition pos)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** | **X** |  |  |  |  |  | | **1** |  |  |  |  |  |  | | **2** |  |  | A |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  |  |  | | **5** |  |  |  |  |  | O |   pos.getRow = 0  pos.getColumn = 0 | **Output:**  whatsAtPos = ‘X’  state of board is unchanged | **Reason:**  This test cause is distinct and unique it returns a player at the lower bounds of the board.  **Function Name:**  testWhatsAtPos\_up\_left |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** | X |  |  |  |  |  | | **1** |  |  |  |  |  |  | | **2** |  |  | A |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  |  |  | | **5** |  |  |  |  |  | **O** |   pos.getRow = 5  pos.getColumn = 5 | **Output:**  whatsAtPos = ‘O’  state of board is unchanged | **Reason:**  This test cause is distinct and unique because it returns the player at the upper bound of the board.  **Function Name:**  testWhatsAtPos\_low\_right |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** | X |  |  |  |  |  | | **1** |  |  |  |  |  |  | | **2** |  |  | A |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  |  |  | | **5** |  |  |  |  |  | O |   pos.getRow = 0  pos.getColumn = 5 | **Output:**  whatsAtPos = ‘ ‘  state of board is unchanged | **Reason:**  This test cause is distinct and unique because it returns an empty space at another upper bound, this time on the upper right corner.  **Function Name:**  testWhatsAtPos\_empty\_up\_right |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  | O | | **1** |  |  |  |  |  |  | | **2** |  |  |  |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  |  |  | | **5** | **X** |  |  |  |  |  |   pos.getRow = 5  pos.getColumn = 0 | **Output:**  whatsAtPos = ‘X’  state of board is unchanged | **Reason:**  This test cause is distinct and unique because it differentiates between two inverted locations. (notice X (5,0) and O (0,5))  **Function Name:**  testWhatsAtPos\_inverse\_lower\_left |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  | **O** | | **1** |  |  |  |  |  |  | | **2** |  |  |  |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  |  |  | | **5** | X |  |  |  |  |  |   pos.getRow = 0  pos.getColumn = 5 | **Output:**  whatsAtPos = ‘O’  state of board is unchanged | **Reason:**  This test cause is distinct and unique because it tests that the right player token will be returned when looking between to pieces that could easily be confused.  **Function Name:**  testWhatsAtPos\_inverse\_upper\_right |

boolean isPlayerAtPos(BoardPosition pos, char player)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  | O | O |  |  | A | | **1** |  |  |  |  |  |  | | **2** |  |  |  |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  | X |  | | **5** | A |  |  |  |  | **X** |   pos.getRow = 5  pos.getColumn = 5  player = ‘X’ | **Output:**  isPlayerAtPos = True  state of board is unchanged | **Reason:**  This test case is distinct and unique because it accounts for the case that the player token is at the given position. Also is a boundary case in which the row and columns are both upper bounds in the lower right corner.  **Function Name:**  testIsPlayerAtPos\_belongs\_to\_player |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  | O | O |  |  | A | | **1** |  |  |  |  |  |  | | **2** |  |  |  |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  | X |  | | **5** | A |  |  |  |  | X |   pos.getRow = 0  pos.getColumn = 0  player = ‘X’ | **Output:**  isPlayerAtPos = False  state of board is unchanged | **Reason:**  This test case is distinct and unique because it accounts for the case that the position passed in may be empty. It also accounts for the lower bound of row and column in the top left corner.  **Function Name:**  testIsPlayerAtPos\_empty\_position |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  | O | O |  |  | **A** | | **1** |  |  |  |  |  |  | | **2** |  |  |  |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  | X |  | | **5** | A |  |  |  |  | X |   pos.getRow = 0  pos.getColumn = 5  player = ‘X’ | **Output:**  isPlayerAtPos = False  state of board is unchanged | **Reason:**  This test case is distinct and unique because it accounts for the case that the position passed in belongs to another player. Additionally, the position is another boundary case in the upper right corner.  **Function Name:**  testIsPlayerAtPos\_wrong\_player |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  | O | O |  |  | A | | **1** |  |  |  |  |  |  | | **2** |  |  |  |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  | X |  | | **5** | **A** |  |  |  |  | X |   pos.getRow = 5  pos.getColumn = 0  player = ‘B’ | **Output:**  isPlayerAtPos = False  state of board is unchanged | **Reason:**  This test case is distinct and unique because it accounts for a player that has not been placed in the board. The position is also a boundary of the board at the lower left corner.  **Function Name:**  testIsPlayerAtPos\_player\_not\_placed |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  | O | O |  |  | A | | **1** |  |  |  |  |  |  | | **2** |  |  |  |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  | X |  | | **5** | A |  |  |  |  | X |   pos.getRow = 3  pos.getColumn = 2  player = ‘W’ | **Output:**  isPlayerAtPos = False  state of board is unchanged | **Reason:**  This test case is distinct and unique because it accounts for a case where the position is empty, and a player has not been placed. It also tests for a position that is not on the boundary of boad.  **Function Name:**  testIsPlayerAtPos\_empty\_and\_player\_not\_placed |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  | O | O |  |  | A | | **1** |  |  |  |  |  |  | | **2** |  |  |  |  |  |  | | **3** |  |  |  |  |  |  | | **4** |  |  |  |  | X |  | | **5** | A |  |  |  |  | X |   pos.getRow = 3  pos.getColumn = 2  player = ‘ ’ | **Output:**  isPlayerAtPos = True  state of board is unchanged | **Reason:**  This test case is distinct and unique because it accounts for a case where the position is empty, and a player token is also. It also tests for a position that is not on the boundary of board.  **Function Name:**  testIsPlayerAtPos\_empty\_and\_player\_is\_empty |

void placeMarker(BoardPosition marker, char player)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  |  |   marker.getRow = 5  marker.getColumn = 5  player = ‘X’ | **Output:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  | **X** | | **Reason:**  This test case is distinct and unique because it places a marker at the lower right corner boundary.  **Function Name:**  testPlaceMarker\_low\_right |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  |  |   marker.getRow = 0  marker.getColumn = 0  player = ‘X’ | **Output:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** | **X** |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  |  | | **Reason:**  This test case is distinct and unique because it places a marker at the upper left corner boundary.  **Function Name:**  testPlaceMarker\_up\_left |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  |  |   marker.getRow = 0  marker.getColumn = 5  player = ‘X’ | **Output:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  | **X** | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  |  | | **Reason:**  This test case is distinct and unique because it places a marker at the upper right corner boundary.  **Function Name:**  testPlaceMarker\_up\_right |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  |  |   marker.getRow = 5  marker.getColumn = 0  player = ‘X’ | **Output:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** | **X** |  |  |  |  |  | | **Reason:**  This test case is distinct and unique because it places a marker at the lower right corner boundary.  **Function Name:**  testPlaceMarker\_low\_left |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  |  |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  |  |   marker.getRow = 2  marker.getColumn = 2  player = ‘B’ | **Output:**  **State:**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **0** | **1** | **2** | **3** | **4** | **5** | | **0** |  |  |  |  |  |  | | **1** | O |  |  |  |  |  | | **2** |  |  | **B** |  | A |  | | **3** |  |  |  |  |  |  | | **4** | X |  |  |  |  |  | | **5** |  |  |  |  |  |  | | **Reason:**  This test case is distinct and unique because it places a marker from a player who has not been placed. Also uses a position which is in the middle and not a boundary.  **Function Name:**  testPlaceMarker\_player\_not\_placed |