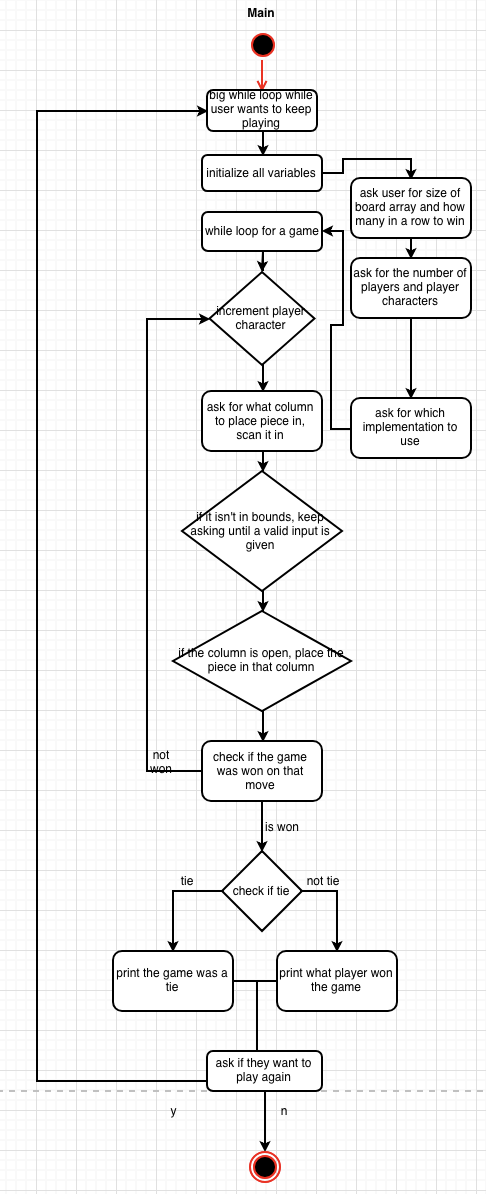
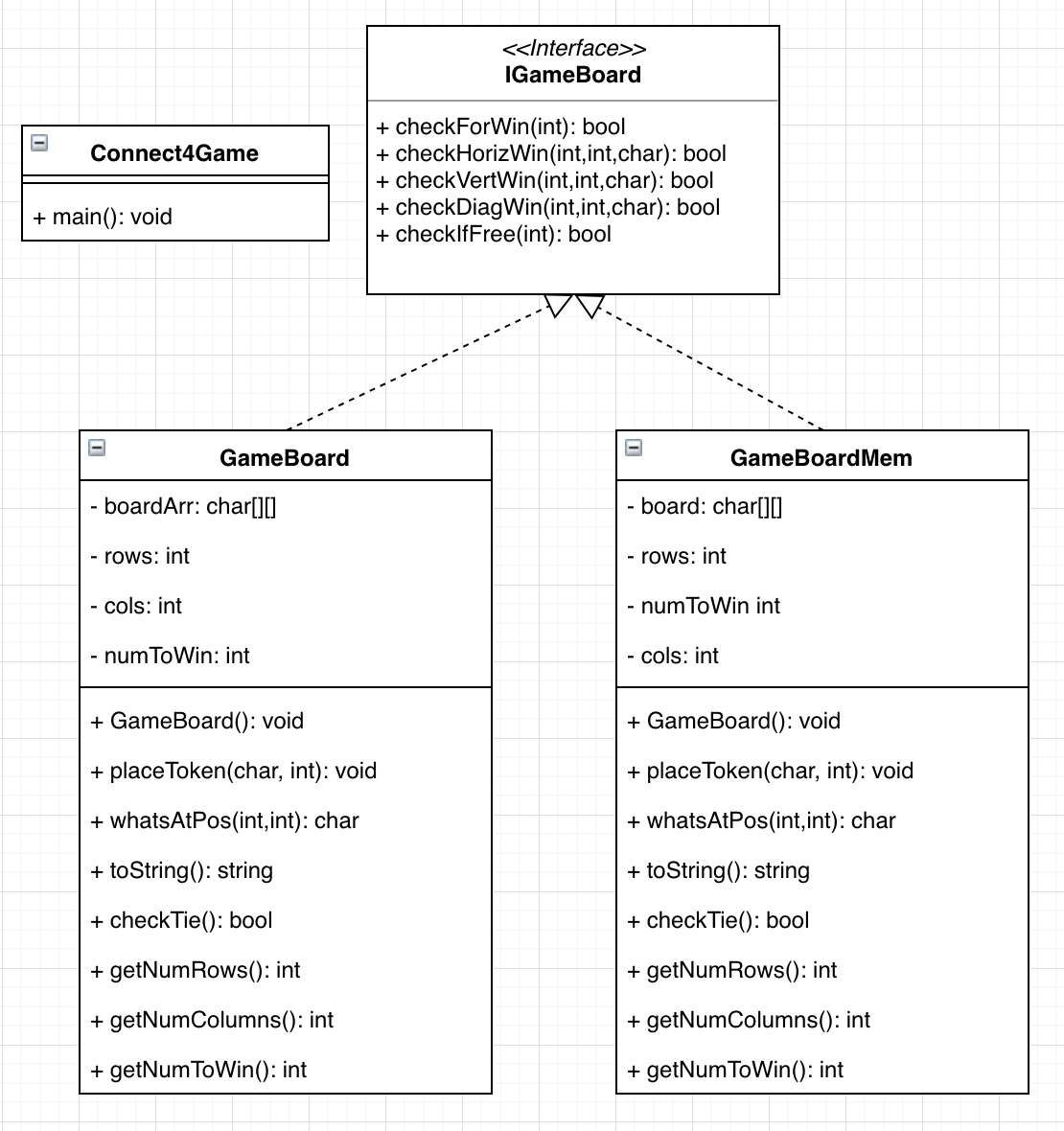
**Homework 4 – Ben Joye**

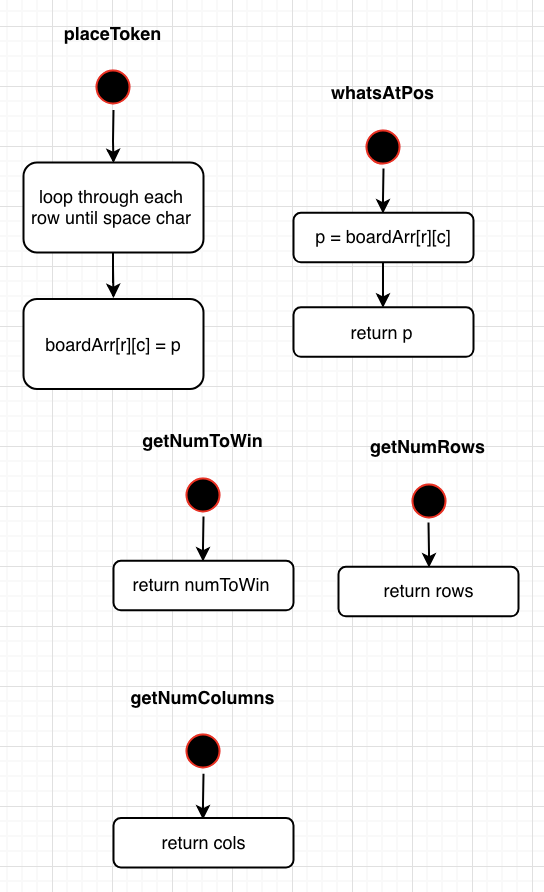
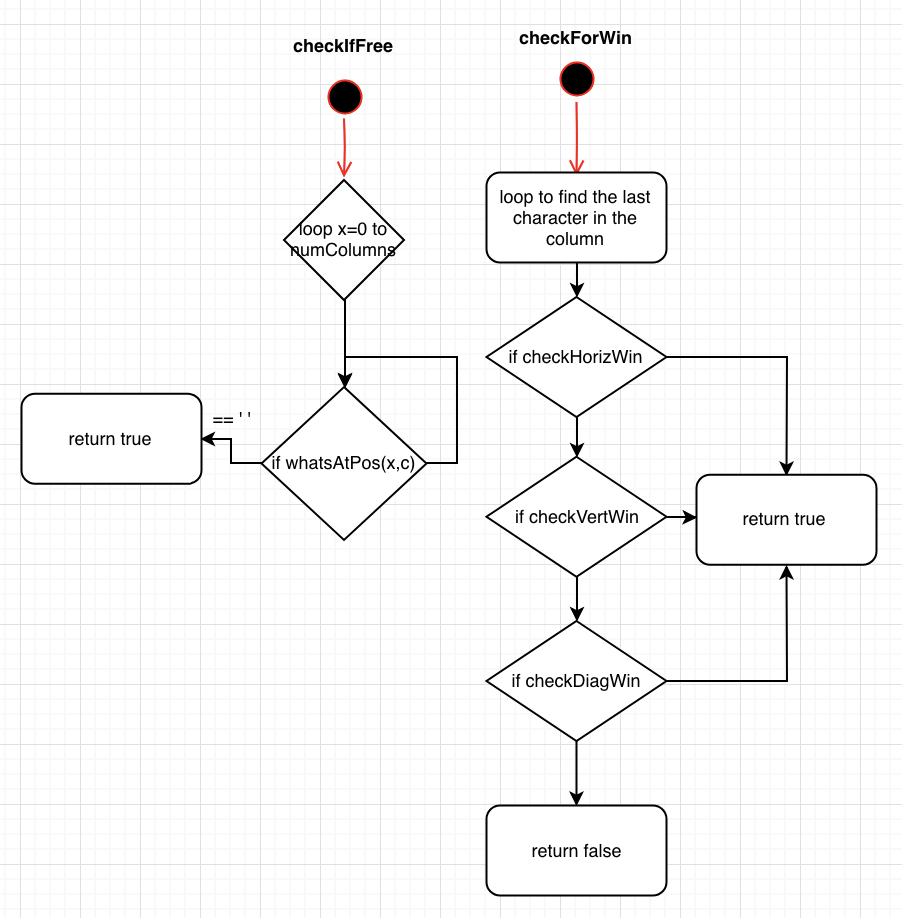
**CPSC 2150**

**Requirements Analysis:**

* **Functional**
  + As a user, I can input what column to place my piece into so that it is stored in the board array
  + As a user, I can see the board array after every turn.
  + As a user, both players will alternate turns.
  + As a user, I can choose to play again so that the game will keep running.
  + As a user, I can input numbers to decide the size of the game board.
  + As a user, I can input numbers to decide the number of pieces in a row you need to win.
  + As a user, I can input an integer to set how many players can play.
  + As a user, I can input characters to set the symbols for each player.
  + As a user, I can input characters to decide which implementation to use.
* **Non-Functional**
  + The system must be able to detect when a player has won.
  + The system must display which players turn it is.
  + The system must keep track of every move and display the board after every turn.
  + The system must handle a board size of up to 100 rows and 100 columns.
  + The system only lets dimensions from 3 to 100 for the board array.

**Design:**

****

****

**Testing:**

**GameBoard()**

*void testIGameBoard\_min()*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State: null | Output State:   |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  |   Empty board | Reason: tests the minimum values for rows and columns |

*void testIGameBoard\_max()*

|  |  |  |
| --- | --- | --- |
| Input State: null | Output State:  100x100 board | Reason: tests the maximum values for rows and columns |

*void testIGameBoard\_minandmax()*

|  |  |  |
| --- | --- | --- |
| Input State: null | Output State:  3x100 board | Reason: tests the minimum and maximum values for rows and columns together |

**Char WhatsAtPos(int r, int c)**

*void testWhatsAtPos\_origin()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | | X |  |  |  |  |  |   R=0, C=0 | Output State:  Return ‘X’  State of the board is unchanged | Reason:  Testing the first spot in the array, could be problematic |

*void testWhatsAtPos\_2ndrow()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | | O |  |  |  |  |  | | X |  |  |  |  |  |   R=1, C=0 | Output State:  Return ‘O’  State of the board is unchanged | Reason:  Testing if there are multiple characters in a column |

*void testWhatsAtPos\_top()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | X |  |  |  |  |  | | X |  |  |  |  |  | | X |  |  |  |  |  | | X |  |  |  |  |  |   R=3, C=0 | Output State:  Return ‘X’  State of the board is unchanged | Reason:  Testing if there are multiple characters in a column |

*void testWhatsAtPos\_3rdrow()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | X |  |  |  |  |  | | X |  |  |  |  |  | | X |  |  |  |  |  |   R=2, C=0 | Output State:  Return ‘X’  State of the board is unchanged | Reason:  Testing if there are more than 2 characters in a column |

*void testWhatsAtPos\_middle()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  | X |  |  | |  |  |  | X |  |  | |  |  |  | X |  |  |   R=2, C=3 | Output State:  Return ‘X’  State of the board is unchanged | Reason:  Testing other columns than the first one |

*void testWhatsAtPos\_topmiddle()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  | X |  |  | |  |  |  | X |  |  | |  |  |  | X |  |  | |  |  |  | X |  |  |   R=3, C=3 | Output State:  Return ‘X’  State of the board is unchanged | Reason:  Testing the top of other columns than the first one |

*void testWhatsAtPos\_topmiddle()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | X | |  |  |  |  |  | X | |  |  |  |  |  | X | |  |  |  |  |  | X |   R=3, C=5 | Output State:  Return ‘X’  State of the board is unchanged | Reason:  Testing the far top corner of the board |

**Void PlaceToken(char p, int c)**

*void testPlaceToken\_origin()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  |   P=X, C=0 | Output State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | | X |  |  |  |  |  | | Reason:  Tests that the user can place in the first spot of the board |

*void testPlaceToken\_opencolumn()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  |   P=X, C=2 | Output State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  | X |  |  |  | | Reason:  Tests that the user can place a token in a different open column |

*void testPlaceToken\_difchars()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  | X |  |  |  |   P=O, C=2 | Output State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  | O |  |  |  | |  |  | X |  |  |  | | Reason:  Tests that the user can place different chars in the same column |

*void testPlaceToken\_toprow()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  | X |  |  |  | |  |  | O |  |  |  | |  |  | X |  |  |  |   P=O, C=2 | Output State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  | O |  |  |  | |  |  | X |  |  |  | |  |  | O |  |  |  | |  |  | X |  |  |  | | Reason:  Tests that the user can place a token in the top row |

*void testPlaceToken\_topcorner()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  | X | |  |  |  |  |  | X | |  |  |  |  |  | X |   P=X, C=5 | Output State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | X | |  |  |  |  |  | X | |  |  |  |  |  | X | |  |  |  |  |  | X | | Reason:  Tests that the user can place a token in the far top corner of the board |

**Boolean CheckIfFree(int c)**

*void testCheckIfFree\_emptyboard()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  |   C=0 | Output State:  Return true;  State of the board is unchanged | Reason:  Empty board could cause problems |

*void testCheckIfFree\_fullcolumn()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  | X |  |  |  | |  |  | X |  |  |  | |  |  | X |  |  |  | |  |  | X |  |  |  |   C=2 | Output State:  Return false;  State of the board is unchanged | Reason:  Makes sure one column fills up |

*void testCheckIfFree\_halffull()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  | X |  |  |  | |  |  | X |  |  |  |   C=2 | Output State:  Return true;  State of the board is unchanged | Reason:  Makes sure you can still place if its half full |

**Boolean CheckHorizWin(int r, int c, char p)**

*void testCheckHorizWin\_bottomrow()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | | X | X | X |  |  |  |   R=0, C=2, P=X | Output State:  Return true; | Reason:  Basic 3 in a row |

*void testCheckHorizWin\_bottomrowdifchars()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | | X | O | X |  |  |  |   R=0, C=2, P=X | Output State:  Return false; | Reason:  3 in row but different chars |

*void testCheckHorizWin\_toprow()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | X | X | X |  |  |  | | O | O | O |  |  |  | | O | O | O |  |  |  | | O | O | O |  |  |  |   R=3, C=2, P=X | Output State:  Return true; | Reason:  Makes sure 3 in a row works on the top row |

*void testCheckHorizWin\_middle()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  | X | X | X |  | |  |  | O | O | O |  | |  |  | O | O | O |  |   R=0, C=2, P=X | Output State:  Return true; | Reason:  Makes sure it works in the middle of the board |

*void testCheckHorizWin\_mixed()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  | O |  |  |  |  | |  | X | X | X |  |  | |  | X | O | O |  |  |   R=1, C=3, P=X | Output State:  Return true; | Reason:  Pieces are all mixed up |

**Boolean CheckVertWin(int r, int c, char p)**

*void testCheckVertWin\_left()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | O |  |  |  |  |  | | O |  |  |  |  |  | | O |  |  |  |  |  |   R=2, C=0, P=O | Output State:  Return true; | Reason:  Easy 3 in a row |

*void testCheckVertWin\_nowin()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  | O |  |  |  |  | |  | O |  |  |  |  | |  | X |  |  |  |  |   R=2, C=1, P=O | Output State:  Return false; | Reason:  3 in a row but different chars |

*void testCheckVertWin\_right()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | O | |  |  |  |  |  | O | |  |  |  |  |  | O | |  |  |  |  |  | X |   R=3, C=5, P=O | Output State:  Return true; | Reason:  Makes sure 3 in a row works up in the top |

*void testCheckVertWin\_middle()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  | O |  |  |  |  | |  | O |  |  |  |  | |  | O |  |  |  |  |   R=2, C=1, P=O | Output State:  Return true; | Reason:  3 in a row works in the middle of the board |

*void testCheckVertWin\_horiz()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | | X | X | X |  |  |  |   R=0, C=2, P=X | Output State:  Return false; | Reason:  Makes sure a horizontal win doesn’t trip it |

**Boolean CheckDiagWin(int r, int c, char p)**

*void testCheckDiagWin\_lefttoright()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  | X |  |  |  | |  | X | X |  |  |  | | X | O | O |  |  |  |   R=2, C=2, P=X | Output State:  Return true; | Reason:  Basic 3 in a row |

*void testCheckDiagWin\_righttoleft()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | X |  |  |  |  |  | | X | X |  |  |  |  | | O | O | X |  |  |  |   R=2, C=0, P=X | Output State:  Return true; | Reason:  3 in a row in the other direction |

*void testCheckDiagWin\_horiz()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | | X | X | X |  |  |  |   R=0, C=2, P=X | Output State:  Return false; | Reason:  Makes sure horizontal win doesn’t trip it |

*void testCheckDiagWin\_vert()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | | X |  |  |  |  |  | | X |  |  |  |  |  | | X |  |  |  |  |  |   R=0, C=2, P=X | Output State:  Return false; | Reason:  Makes sure vertical win doesn’t trip it |

**Boolean CheckTie()**

*void testCheckTie\_notie()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  | O |  |  |  |  | |  | X | X | X |  |  | |  | X | O | O |  |  | | Output State:  Return false; | Reason:  Random board situation |

*void testCheckTie\_fullboard()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | X | X | X | X | X | X | | O | O | O | O | O | O | | O | O | O | O | O | O | | O | O | O | O | O | O | | Output State:  Return true; | Reason:  Board completely full |

*void testCheckTie\_onecolfull()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | O | |  |  |  |  |  | O | |  |  |  |  |  | O | |  |  |  |  |  | X | | Output State:  Return false; | Reason:  Makes sure one column being full doesn’t trip it |

*void testCheckTie\_onerowfull()*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input State:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | | X | X | X | X | X | X | | Output State:  Return false; | Reason:  Makes sure a whole row being full doesn’t trip it |

**Deployment:**

* Type make to compile the program
* Type make run to run the program