Testing

1. checkConstructorRows – This case creates a 5x5 GameBoard, and checks to see that the rows = 5. I created this to test the rows part of the constructor, and it is distinct because it is the only test that does such.
2. checkConstructorColumns – This case creates a 5x5 GameBoard, and checks to see that the columns = 5. I created this to test the columns part of the constructor, and it is distinct because it is the only test that does such.
3. checkConstructorNum – This case creates a 5x5 GameBoard with the number to win 5 in a row, and checks to see that the num = 5. I created this to test the num to win part of the constructor, and it is distinct because it is the only test that does such.
4. checkIfFreeBlank – This test creates a 5x5 Gameboard without placing any tokens, and checks to see that the 4th column is free. I chose this case because I wanted to confirm that the checkIfFree function works on an empty board like it should, and it's distinct because it's the only checkIfFree test that looks at an empty board.
5. checkIfFreeFullFirstColumn – This test works with a 5x5 GameBoard that is completely full, and checks that the first column is full. I chose this case and it is distinct because I wanted to check that the checkIfFree worked when called on the first row (edge) when the board was full.
6. checkIfFreeFullLastColumn – This test works with a 5x5 GameBoard that is completely full, and checks that the last column is full. I chose this case and it is distinct because I wanted to check that the checkIfFree worked when called on the last row (edge) when the board was full.
7. checkHorizWinEmpty – This test works with a 5x5 GameBoard with 5 in a row to win and checks that there is no horizontal win when the board is empty. I chose this to test that my checkHorizWin worked with an empty board, and it is distinct because it's the only horizontal win test with an empty board.
8. checkHorizWinTrue – This test works with a 4x4 GameBoard with 3 in a row to win, and places an X in columns 0, 1, and 2. I chose this test just to see if my checkHorizWin would work with a very simple win, and it's distinct because it only has 3 of one character and no others.
9. checkHorizWinFullFalse – This test works with a 4x4 GameBoard with 4 in a row to win, and checks that a GameBoard with alternating letters won't return true with a horizontal win. I chose this to see if my checkHorizWin would work with a tie, and it's distinct because it's the only false tie checking horizontals.
10. checkHorizWinFullTrue – This test works with a 4x4 GameBoard with 4 in a row to win, and checks that a full GameBoard with 4 X's in a row horizontally will return true with a horizontal win. I chose this to see if my checkHorizWin would work with a full board, and it's distinct because it's the only full board with a horizontal win.
11. checkHorizWinMiddleBlank – This case works with a 4x4 GameBoard with 3 in a row to win, and places X's in columns 0, 1, and 3. I chose this to test if my checkHorizWin would return false if there's 3 of the same characters in the same row, but a blank in between them and it's distinct because it's the only horizontal test that uses a blank in the middle.

12. checkVertWinEmpty – This test works with a 4x4 GameBoard with 4 in a row to win and checks that there is no vertical win when the board is empty. I chose this to test that my checkVertWin worked with an empty board, and it is distinct because it's the only vertical win test with an empty board.
13. checkVertWinFullFalse – This test works with a 4x4 GameBoard with 4 in a row to win, and checks that a GameBoard with alternating letters won't return true with a vertical win. I chose this to see if my checkVertWin would work with a tie, and it's distinct because it's the only false tie checking verticals.
14. checkVertWinFullTrue – This test works with a 4x4 GameBoard with 3 in a row to win, and checks that a full GameBoard with 3 X's in a row vertically will return true with a vertical win. I chose this to see if my checkVertWin would work with a full board, and it's distinct because it's the only full board with a vertical win.
15. checkVertWinTrue – This test works with a 4x4 GameBoard with 3 in a row to win, and places an X in rows 3, 2, and 1 of column 0. I chose this test just to see if my checkVertWin would work with a very simple win, and it's distinct because it only has 3 of one character and no others.
16. checkVertWinSpace - This case works with a 4x4 GameBoard with 3 in a row to win, and places X's in rows 0, 1, and 3 of column 0, with an O in row 2. I chose this to test if my checkVertWin would return false if there's 3 of the same characters in the same row, but a different character in between them, and it's distinct because it's the only vertical test that uses a blank in the middle.
17. checkDiagWinEmpty – This test works with a 4x4 GameBoard with 4 in a row to win and checks that there is no diagonal win when the board is empty. I chose this to test that my checkDiagWin worked with an empty board, and it is distinct because it's the only diagonal win test with an empty board.
18. checkDiagWinTrue - This test works with a 4x4 GameBoard with 3 in a row to win, and places an X diagonally from GB[3, 0] to GB[1, 2]. I chose this test just to see if my checkDiagWin would work with a very simple win, and it's distinct because it only has 3 of one character and no others.
19. checkDiagWinFullFalse - This test works with a 4x4 GameBoard with 4 in a row to win, and checks that a GameBoard with alternating letters won't return true with a diagonal win. I chose this to see if my checkDiagWin would work with a tie, and it's distinct because it's the only false tie checking diagonals.
20. checkDiagWinFullTrue - This test works with a 4x4 GameBoard with 4 in a row to win, and checks that a full GameBoard with 4 X's in a row diagonally from GB[3, 0] to GB[0 , 3], and everything else an O, will return true with a diagonal win. I chose this to see if my checkDiagWin would work with a full board, and it's distinct because it's the only full board with a diagonal win.
21. checkDiagWinSpace - This case works with a 4x4 GameBoard with 3 in a row to win, and places X's in GB spots [3, 0], [2, 1], and [0, 3] . I chose this to test if my checkDiagWin would return false if there's 3 of the same characters diagonally, but a space in between them, and it's distinct because it's the only diagonal test that uses a blank in the middle.

22. checkDiagWinReverse – This case works with a 4x4 GameBoard with 4 in a row to win, and places X's in GB Spots [0, 0], [1, 1], [2, 2], and [3, 3] to check if checkDiagWin will return true. I chose this case because all my other tests had been going the opposite direction, and this case is distinct because it's the only diagonal test in that direction.
23. checkDiagWinBigBoard – This case works with a 25x25 GameBoard with 25 in a row to win, and places X's in GB spots [24, 0] to [0, 24], and checks that there is a diagonal win. I chose this test because I wanted to see if my checkDiagWin worked on a bigger board, and this test is distinct because it's the only diagonal test with a board this big.
24. checkDiagWinTallNarrowBoard – This test works with a 3x100 GameBoard with 3 in a row to win, and places X's from [99, 0] to [97, 2] to confirm there is a diagonal win. I chose this test because I had only been working with square boards and wanted to test my checkDiagWin on a rectangular board, and this test is distinct because it's the only diagonal test on a rectangular board.
25. checkTieEmpty – This test works with a 4x4 GameBoard with 4 in a row to win, and checks that an empty board won't return a tie. I chose this case to make sure my tie algorithm wouldn't show a tie for an empty board, and this test is distinct because it's the only tie test with an empty board.
26. checkTieTrue - This test works with a 4x4 GameBoard with 4 in a row to win, and checks that a GameBoard with two alternating letters will return true with a tie. I chose this to see if my checkTie would work with a true tie, and it's distinct because it's the only true tie with two players.
27. checkTieThreePlayers - This test works with a 4x4 GameBoard with 4 in a row to win, and checks that a GameBoard with three alternating letters will return true with a tie. I chose this to see if my checkTie would work with a true tie, and it's distinct because it's the only true tie with three players.
28. checkTieFalse – This test works with a 4x4 GameBoard with 3 in a row to win, and places 3 X's in column 0 to test a false tie. I chose this to check that my checkTie wouldn't work on a vertical win, and it's distinct because it's the only checkTie test with only 3 tokens on the board.
29. whatsAtPosEmpty – This test works with a 4x4 GameBoard with 4 in a row to win, and checks that the token at [3, 0] == '\u0000'. I chose this to see if my whatsAtPos works with the null character because that was an essential part of my toString method, and it's distinct because it's the only whatsAtPos test with an empty board.
30. whatsAtPos2 - This test works with a 4x4 GameBoard with 4 in a row to win, and places an X at [3, 0], then checks that there is indeed an X in that spot. I chose this test to see if my whatsAtPos worked at the bottom left of the board, and it's distinct because it's the only test that looks at the bottom left spot.
31. whatsAtPos3 - This test works with a 4x4 GameBoard with 4 in a row to win, and places an X, Y, X, Y in column 0, then checks to see that whatsAtPos(0, 0) indeed is a Y. I chose this case and it is distinct because it is the only one that checks the top of a full column 0.
32. whatsAtPos4 – This test works with a 100x100 GameBoard and places a Q in spot [99,99]. I chose this test and it is distinct because it is the only test that uses a 100x100

GameBoard and checks the bottom right, and I wanted to test my whatsAtPos at the board's max at an edge position.

33. whatsAtPos5 – This test works with a 100x100 GameBoard and places a Q in spot [99,27]. I chose this test and it is distinct because it is the only whatsAtPos test that uses the bottom middle of the board, I just wanted to pick a random spot in the middle because I had only been checking edges.
34. whatsAtPos6 – This test works with a 4x4 GameBoard that has alternating X and O tokens to fill it. I chose this test and it is distinct because it is the only whatsAtPos test that looks at a full board with alternating characters.
35. whatsAtPos7 – This test works with a 4x4 GameBoard filled with X's, then checks that there is indeed an X at spot [2, 2]. I chose this test and it is distinct because it's the only whatsAtPos test that works with a full board of one character.
36. placeToken1 - This test works with a 4x4 GameBoard that places an X in column 0, then checks that the X is put at the bottom of column 0. I chose this test and it is distinct because it is the only placeToken test that checks a single token in column 0.
37. placeToken2 - This test works with a 4x4 GameBoard that places an X in column 3, then checks that the X is put at the bottom of column 3. I chose this test and it is distinct because it is the only placeToken test that checks a single token in column 3.
38. placeToken2 - This test works with a 4x4 GameBoard that places 3 X's and then a Y in column 0, then checks that the Y is put at the top of column 0. I chose this test and it is distinct because it is the only placeToken test that checks the top of column 0 with two characters.
39. placeToken4 – This test works with a 4x4 GameBoard and places an X in column 0, then a Y in column 1 and a Z in column 1, then checks that there is a Z at [2, 1]. I chose this test and it is distinct because it is the only test that uses 3 characters on a 4x4 board.
40. placeToken5 – This test works with a 10x10 GameBoard that places an Y, X, and Z in column 9, then checks that there is a Y at [9, 9]. I chose this case and it is distinct because it's the only placeToken test that checks the bottom of column 9 on a 10x10 board.