Project 3

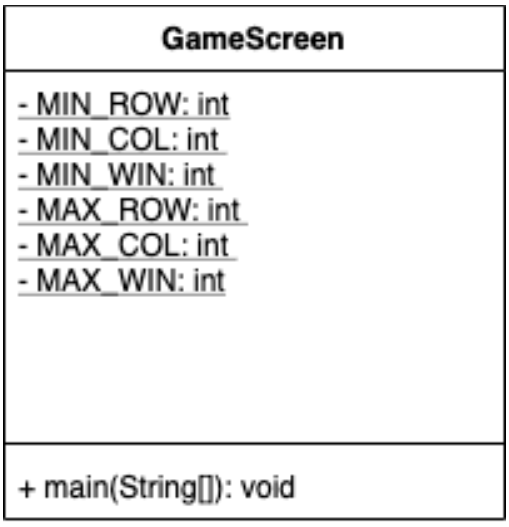
- Abstract ○ In this assignment, we will be designing an extended version of Connect 4 Game that we will run with a command line interface. However, we will be creating an extended version of this game for a user defined size board and requiring user defined number in a row either horizontally, vertically, or diagonally to win.
- Requirements Analysis
- ○ Functional Requirements
    - As a user, I can select an available column, in order to make a move of my choice.
    - As a user, I will know if I win after every move, to accurately know the status of the game.
    - As a user, I can win vertically, horizontally, or diagonally, so I know the basic rules of the game.
    - As a user, I will know when the board is full, the game has resulted in a Tie.
    - As a user, I can set the number of tokens needed in a row to win As a
    - user, I can start a new game immediately after a win or draw without exiting the program, to keep a smooth gameplay.
    - As a user, I can stop playing after each game concludes, so that I'm not stuck in the program.
    - As a user, I will know whether my move is valid, so I know my turn will not be wasted if I make a mistake.
    - As a user, I know after my turn is completed it will switch to the next player automatically, so the flow of the game is not interrupted.
    - As a user, I will be able to see the board after every turn, to be able to pick the best available position.
    - As a user, I will be able to define how many rows are on the board so that I can play on a board of the size of my choosing.
    - As a user, I will be able to define how many columns are on the board so that I can play on a board of the size of my choosing.
    - As a user, I can select how many players are in the game, so that I can play with more than one other person.
    - As a user, I will be able to select between a memory efficient game and a fast game, so that I may play in the style of my liking.
    - As a user, I will be able to select the character I want to play with, so that I am not restricted to just one character forever.

○ Non-functional Requirements

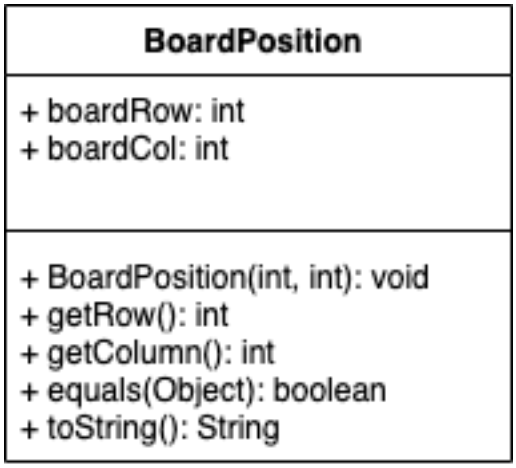
- Must compile on the school of computing servers.
- Board must within bounds of 100x100.
- Number of tokens to win must not exceed the width or height of the
- board.
  Number of tokens to win must not exceed 25.
- The (0,0) position must be at the bottom left of the board.
- The (99,99) position must be the top right of the board.
- The first player to choose a character will always go first.
- Must have at least 2 players, on the same computer.
- Players may choose a letter to play with
- Must be written in Java
- Must be compiled in Java

- How to use the makefile
  - type "`make`", "`make run`" to start running your code and
  - "`makeclean`" to remove any compiled (`.class`) files.

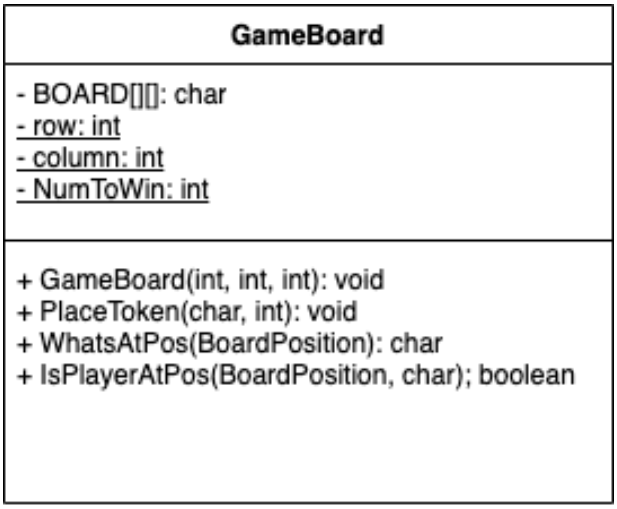
- Design
  - GameScreen



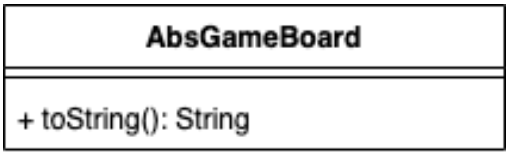
  - BoardPosition



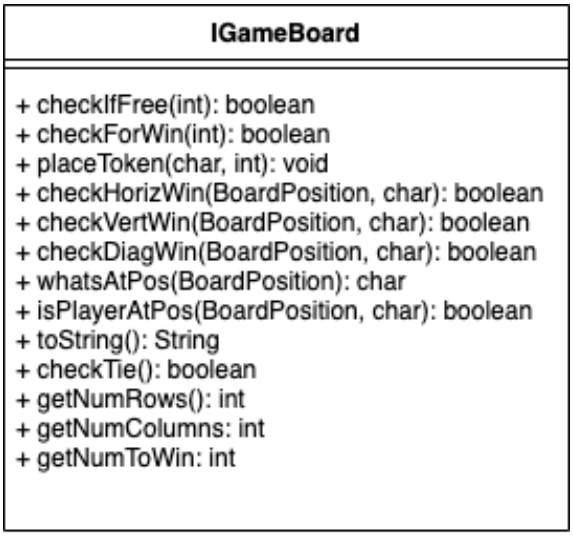
- Gameboard



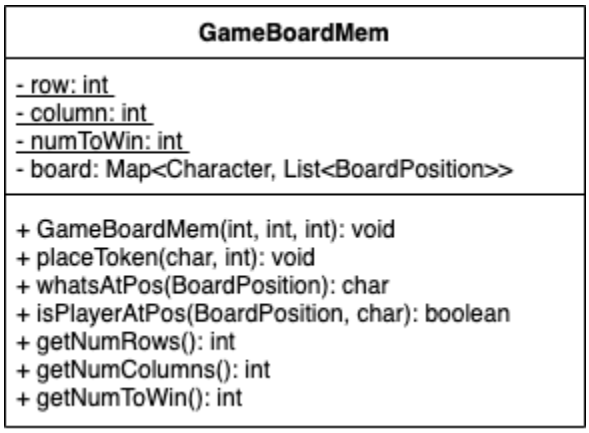
- AbsGameBoard



- IGameBoard

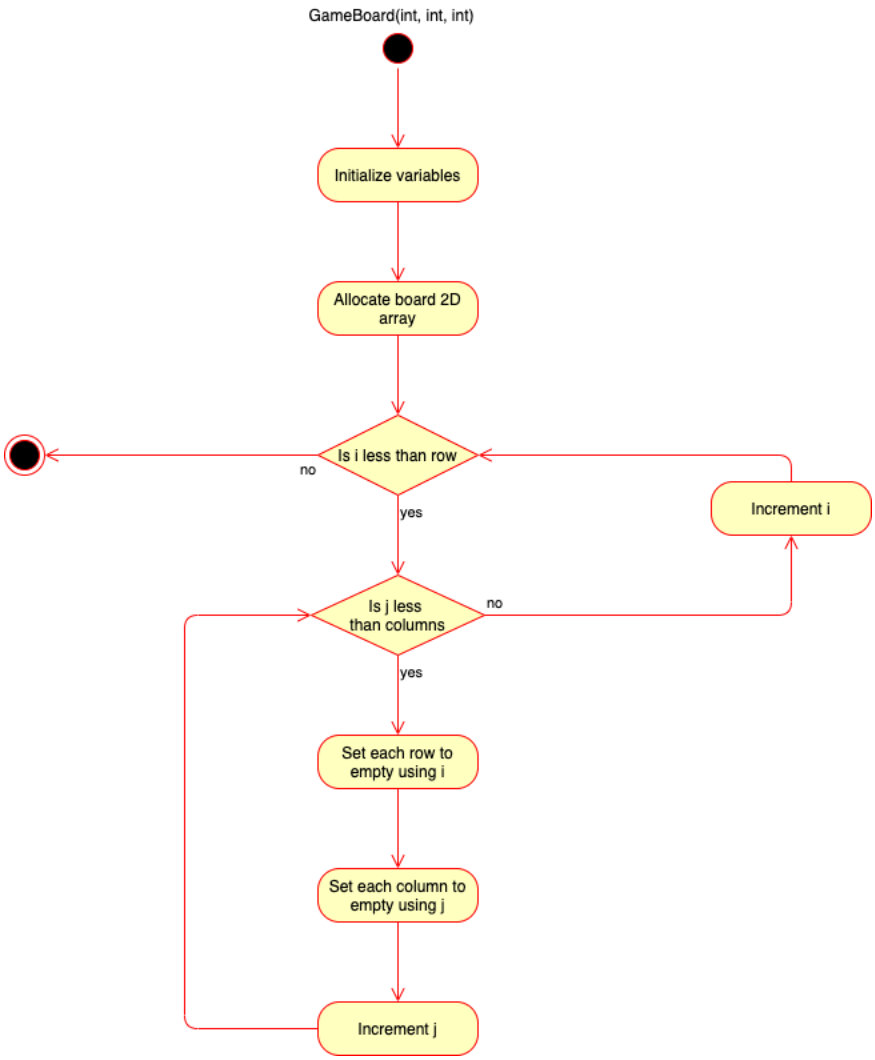


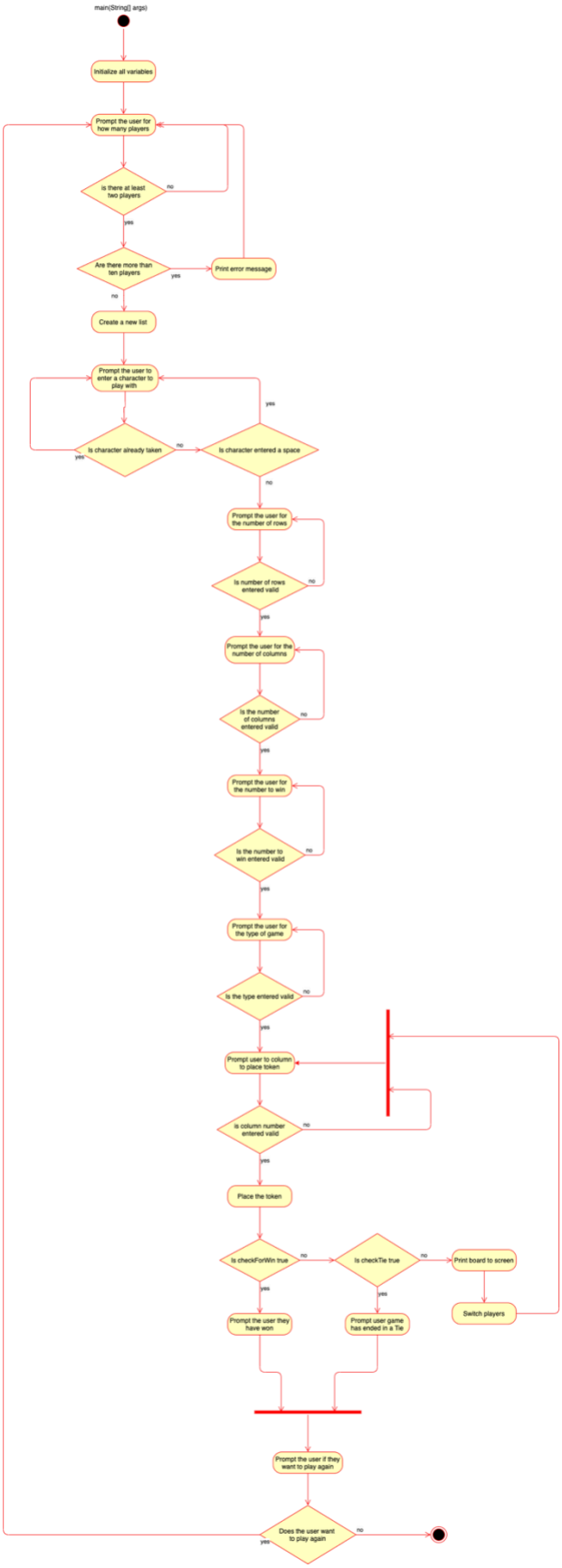
- GameBoardMem

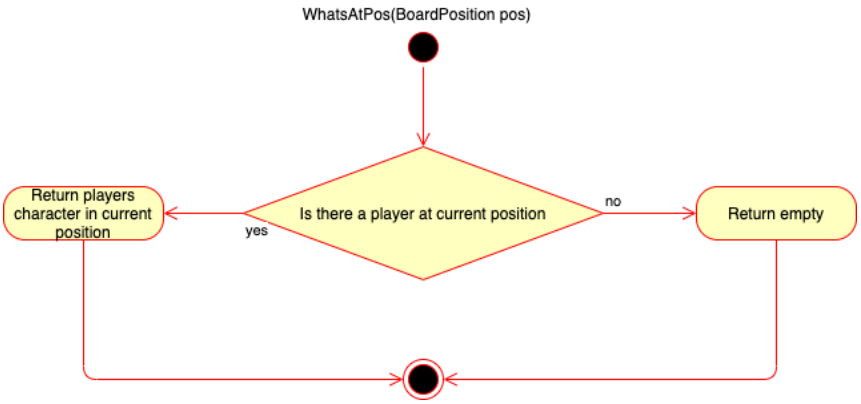


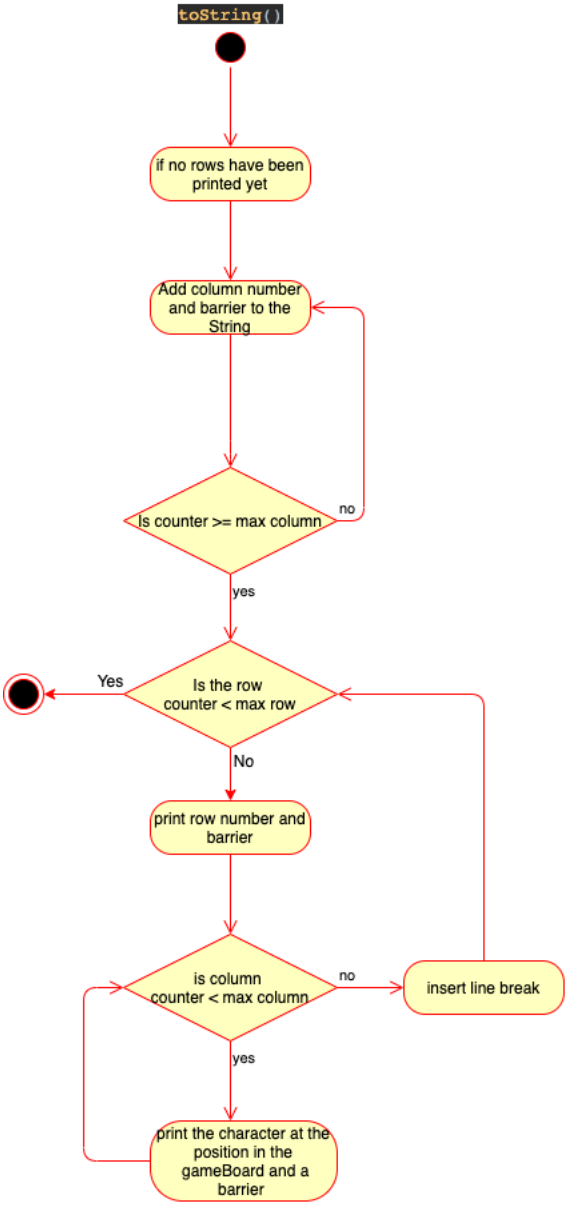
- **Activity Diagrams**

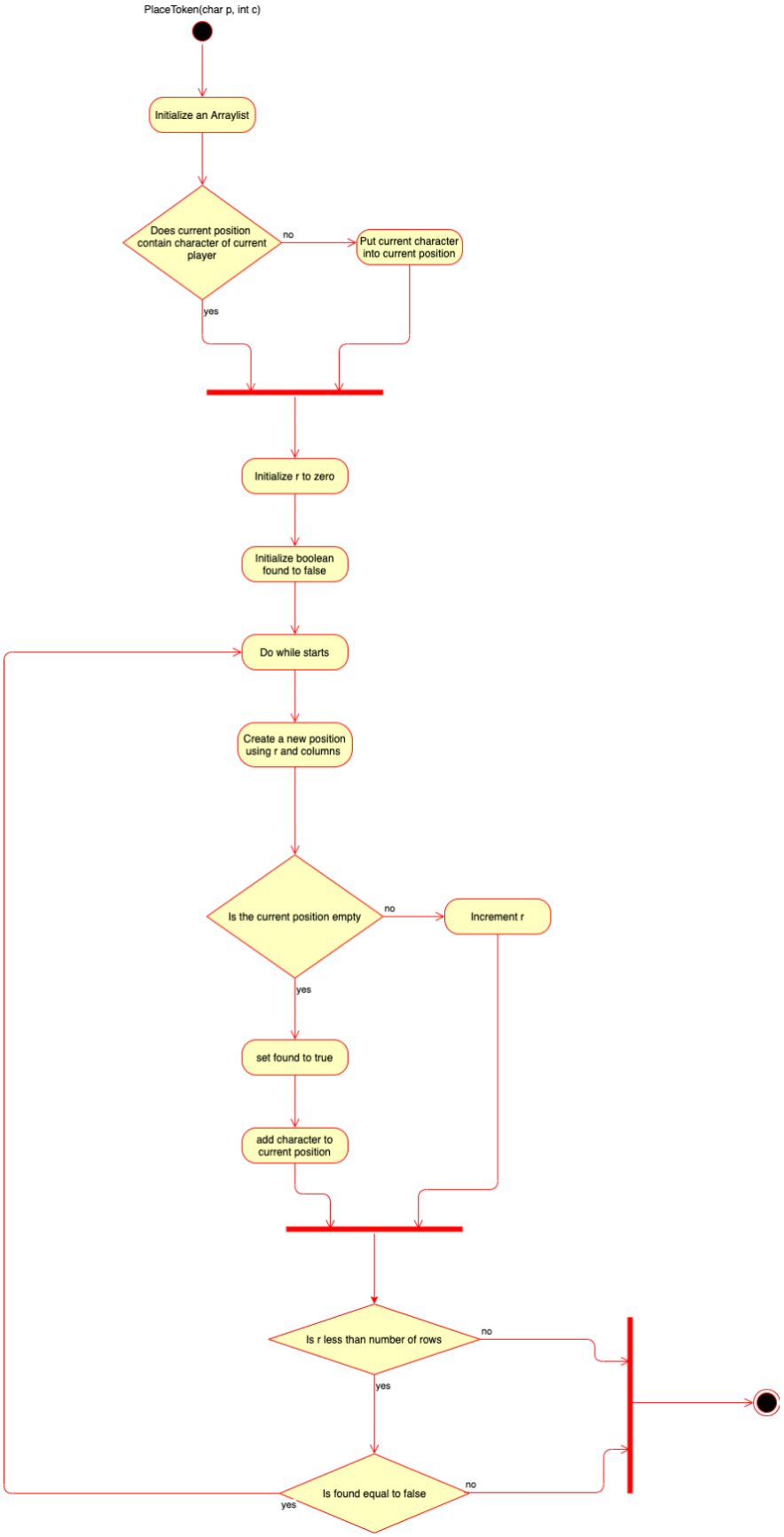
Constructor

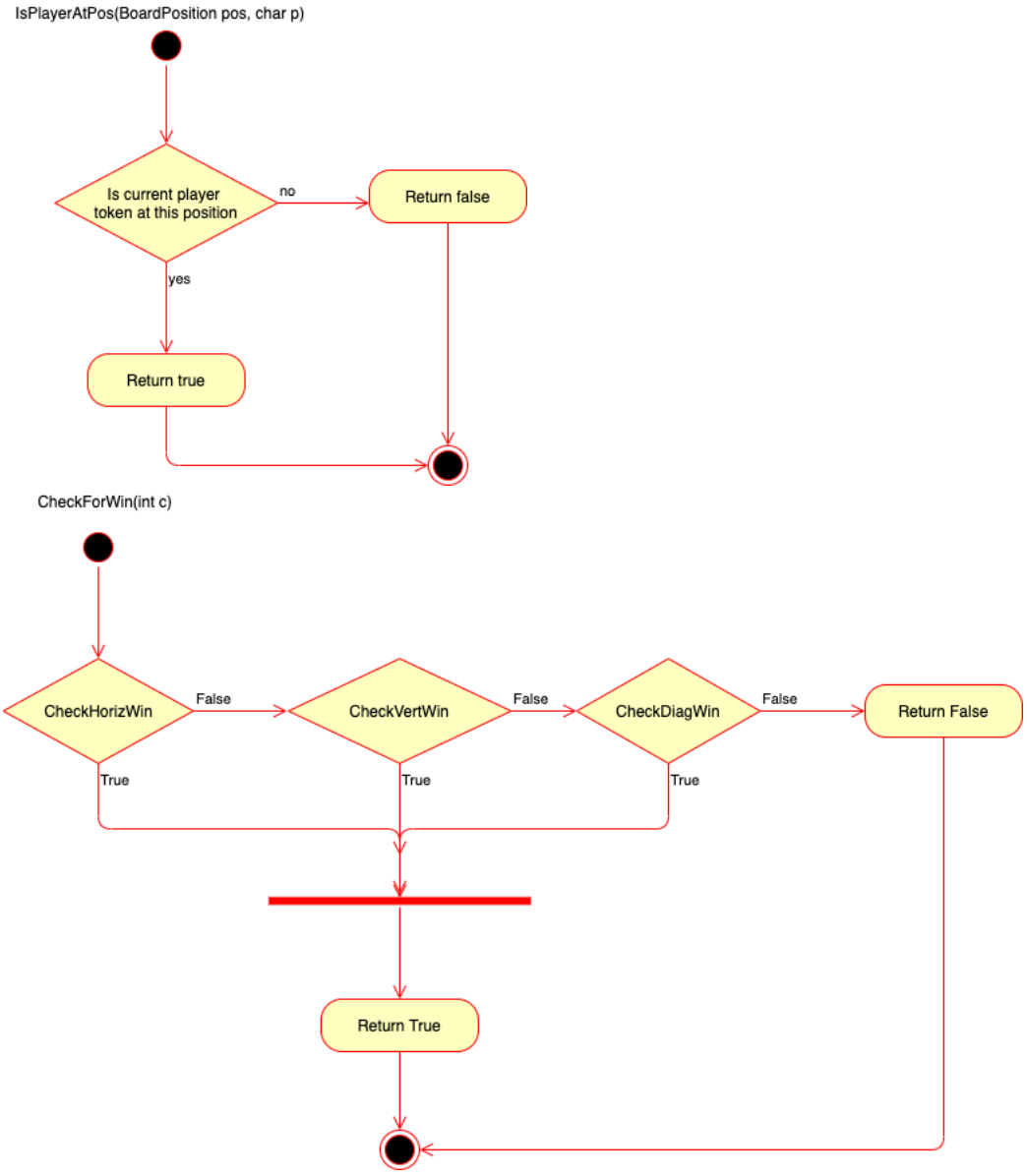

GameBoard(int, int, int)
Initialize variables
Allocate board 2D array
Is i less than row
no
yes
Increment i
Is j less than columns
no
yes
Set each row to empty using i
Set each column to empty using j
Increment j

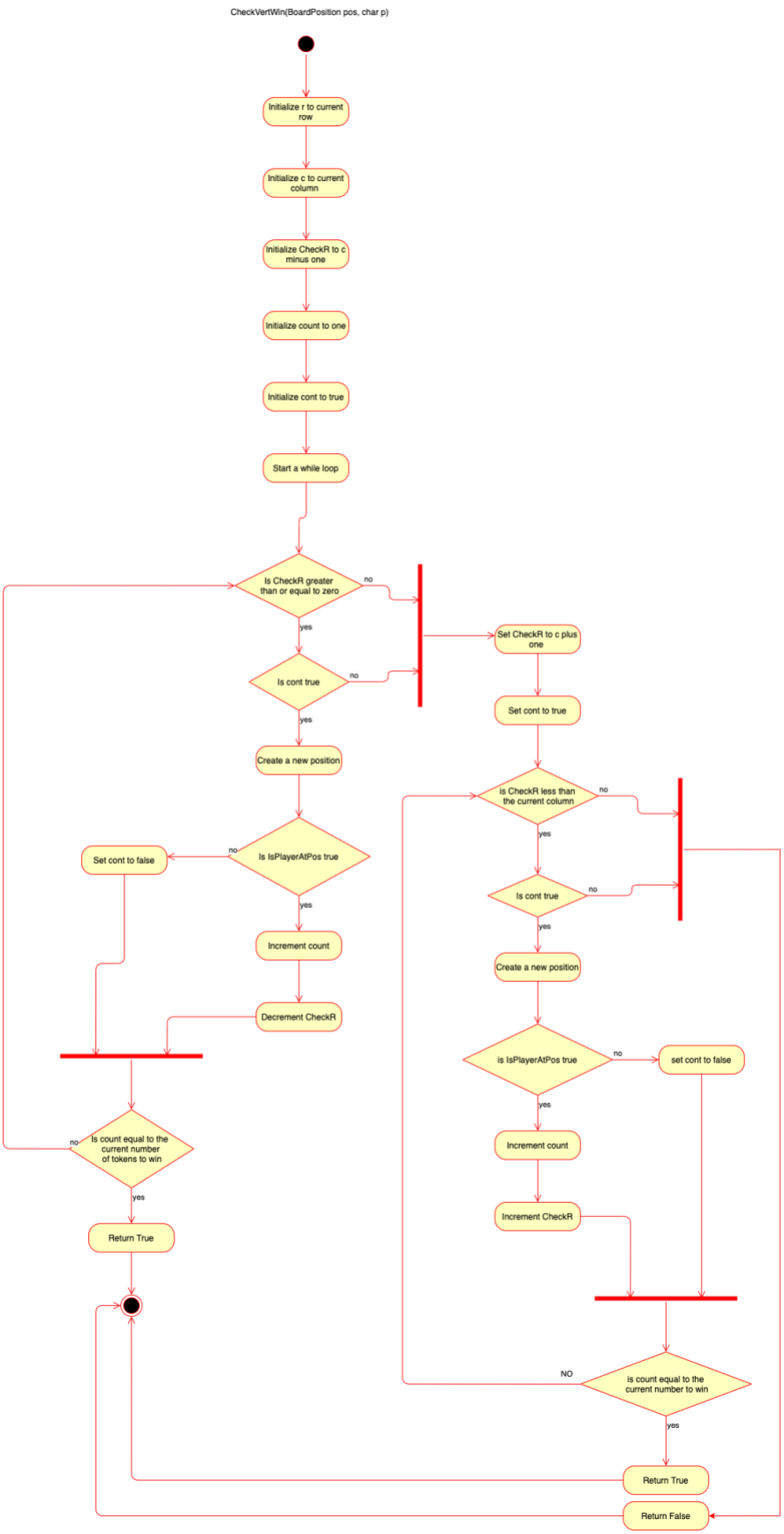

main(String[] args)
Initialize all variables
Prompt the user for how many players
is there at least two players
no
yes
Are there more than ten players
yes
Print error message
no
Create a new list
Prompt the user to enter a character to play with
yes
Is character already taken
yes
no
Is character entered a space
no
Prompt the user for the number of rows
Is number of rows entered valid
no
yes
Prompt the user for the number of columns
Is the number of columns entered valid
no
yes
Prompt the user for the number to win
Is the number to win entered valid
no
yes
Prompt the user for the type of game
Is the type entered valid
no
yes
Prompt user to column to place token
is column number entered valid
no
yes
Place the token
Is checkForWin true
no
yes
Prompt the user they have won
Is checkTie true
no
yes
Prompt user game has ended in a Tie
Print board to screen
Switch players
Prompt the user if they want to play again
Does the user want to play again
yes
no

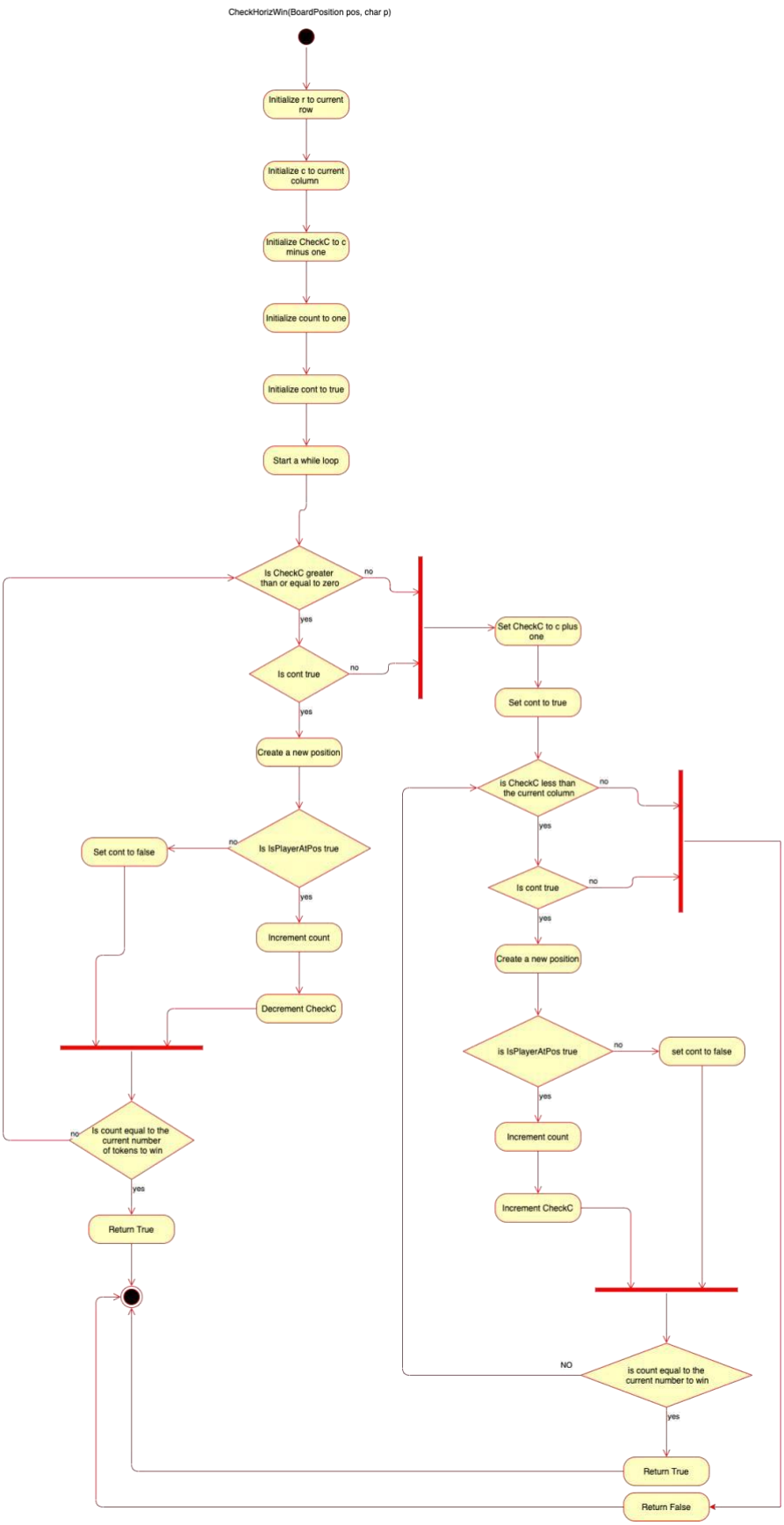

WhatsAtPos(BoardPosition pos)
Is there a player at current position
yes
Return players character in current position
no
Return empty

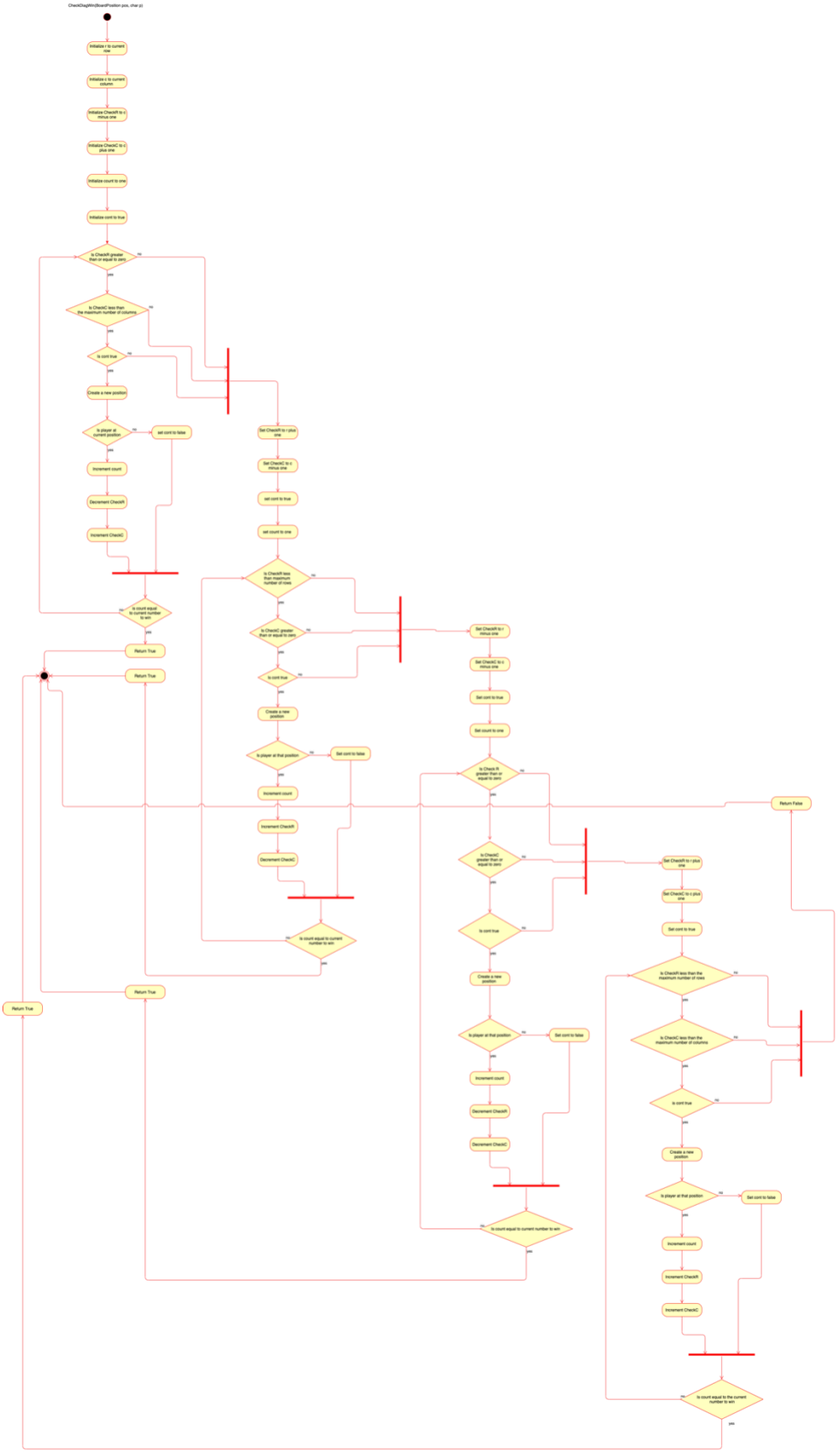

toString()
if no rows have been printed yet
Add column number and barrier to the String
Is counter >= max column
no
yes
Yes
Is the row counter < max row
No
print row number and barrier
is column counter < max column
no
insert line break
yes
print the character at the position in the gameBoard and a barrier

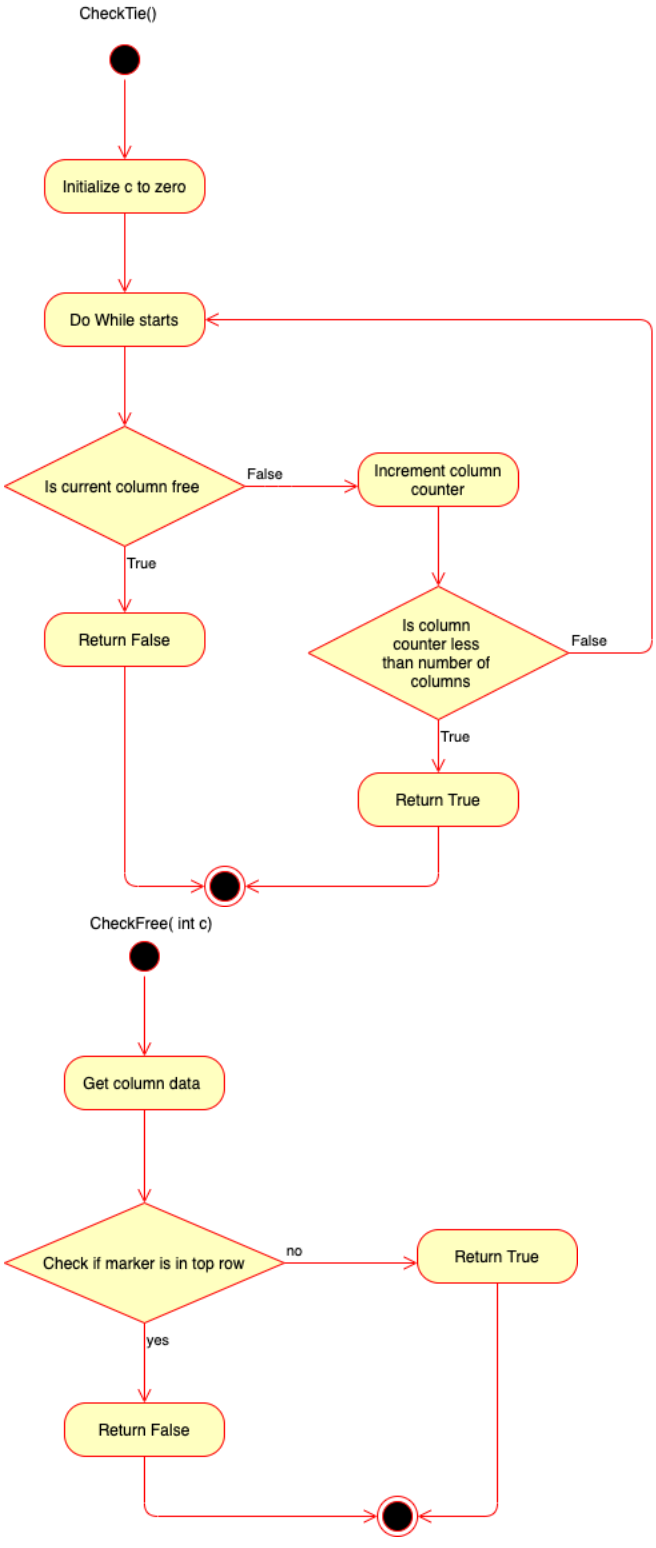

PlaceToken(char p, int c)
Initialize an Arraylist
Does current position contain character of current player
no
Put current character into current position
yes
Initialize r to zero
Initialize boolean found to false
Do while starts
Create a new position using r and columns
Is the current position empty
no
Increment r
yes
set found to true
add character to current position
Is r less than number of rows
no
yes
Is found equal to false
no
yes


IsPlayerAtPos(BoardPosition pos, char p)
Is current player token at this position
no
Return false
yes
Return true
CheckForWin(int c)
CheckHorizWin
False
CheckVertWin
False
CheckDiagWin
False
Return False
True
True
True
Return True


CheckVertWin(BoardPosition pos, char p)
Initialize r to current row
Initialize c to current column
Initialize CheckR to c minus one
Initialize count to one
Initialize cont to true
Start a while loop
Is CheckR greater than or equal to zero
no
yes
Is cont true
no
yes
Create a new position
Is IsPlayerAtPos true
no
Set cont to false
yes
Increment count
Decrement CheckR
Is count equal to the current number of tokens to win
no
yes
Return True
Set CheckR to c plus one
Set cont to true
is CheckR less than the current column
no
yes
Is cont true
no
yes
Create a new position
is IsPlayerAtPos true
no
set cont to false
yes
Increment count
Increment CheckR
is count equal to the current number to win
NO
yes
Return True
Return False


CheckHorizWin(BoardPosition pos, char p)
Initialize r to current row
Initialize c to current column
Initialize CheckC to c minus one
Initialize count to one
Initialize cont to true
Start a while loop
Is CheckC greater than or equal to zero
no
yes
Is cont true
no
yes
Create a new position
Is IsPlayerAtPos true
no
Set cont to false
yes
Increment count
Decrement CheckC
Is count equal to the current number of tokens to win
no
yes
Return True
Set CheckC to c plus one
Set cont to true
is CheckC less than the current column
no
yes
Is cont true
no
yes
Create a new position
is IsPlayerAtPos true
no
set cont to false
yes
Increment count
Increment CheckC
is count equal to the current number to win
NO
yes
Return True
Return False


CheckDiagWin(BoardPosition pos, char p)
Initialize r to current row
Initialize c to current column
Initialize CheckR to r minus one
Initialize CheckC to c plus one
Initialize count to one
Initialize cont to true
Is CheckR greater than or equal to zero
Is CheckC less than the maximum number of columns
Is cont true
Create a new position
Is player at current position
set cont to false
Increment count
Decrement CheckR
Increment CheckC
Is count equal to current number to win
Return True
Return True
Return True
Return True
Set CheckR to r plus one
Set CheckC to c minus one
set cont to true
set count to one
Is CheckR less than maximum number of rows
Is CheckC greater than or equal to zero
Is cont true
Create a new position
Is player at that position
Set cont to false
Increment count
Increment CheckR
Decrement CheckC
Is count equal to current number to win
Set CheckR to r minus one
Set CheckC to c minus one
Set cont to true
Set count to one
Is CheckR greater than or equal to zero
Is CheckC greater than or equal to zero
Is cont true
Create a new position
Is player at that position
Set cont to false
Increment count
Decrement CheckR
Decrement CheckC
Is count equal to current number to win
Return False
Set CheckR to r plus one
Set CheckC to c plus one
Set cont to true
Is CheckR less than the maximum number of rows
Is CheckC less than the maximum number of columns
Is cont true
Create a new position
Is player at that position
Set cont to false
Increment count
Increment CheckR
Increment CheckC
Is count equal to the current number to win


CheckTie()
Initialize c to zero
Do While starts
Is current column free
False
Increment column counter
True
Return False
Is column counter less than number of columns
False
True
Return True
CheckFree(int c)
Get column data
Check if marker is in top row
no
Return True
yes
Return False

- Testing
  - Constructor

Input:	Output:	Reason:
Row = 4 Col = 4 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it initializes each parameter to minimal allowable number Function: public void Constructor Test1

Input:	Output:	Reason:
Row = 5 Col = 5 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>4</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it tests when the number to win is at its max limit

Input:	Output:	Reason:
Row = 10 Col = 15 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>15</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it tests when the rows and columns are uneven.

- CheckIfFree

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks if all rows and columns are free

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks if an entire column is full.

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>1</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>2</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>3</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>…</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>9</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks if half the game board is full

- CheckHorizWin

<table>
<tr>
<td>Input:

Row = 10
Col = 10
Win = 4</td>
<td>Output:

<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td>x</td><td>x</td><td>x</td><td></td><td></td><td></td></tr>
</table>
</td>
<td>Reason:

This test case is unique and distinct because it checks if horizontal win does not return true after only three tokens are placed.</td>
</tr>
</table>

<table>
<tr>
<td>Input:

Row = 10
Col = 10
Win = 4</td>
<td>Output:

<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>o</td><td>o</td><td>o</td><td>o</td><td></td><td></td></tr>
<tr><td>9</td><td>X</td><td>X</td><td>X</td><td>B</td><td></td><td></td></tr>
</table>
</td>
<td>Reason:

This test case is unique and distinct because it checks if horizontal win is working in the second column</td>
</tr>
</table>

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td>g</td><td>g</td><td>g</td><td>G</td><td></td><td></td></tr><tr><td>3</td><td>R</td><td>r</td><td>r</td><td>O</td><td></td><td></td></tr><tr><td>…</td><td>r</td><td>r</td><td>R</td><td>O</td><td></td><td></td></tr><tr><td>9</td><td>r</td><td>r</td><td>r</td><td>O</td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks if horizontal win is working in the first four rows

Input:	Output:	Reason:
Row = 5 Col = 5 Win = 3	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>4</td><td>0</td><td>0</td><td>0</td><td>0</td><td></td></tr></table>	This test case is unique and distinct because it checks if horizontal win is working in the first four rows

- CheckVertWin

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 4	<table><tr><td></td><td>0</td><td>…</td><td>7</td><td>8</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td>X</td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td>X</td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>X</td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks if vertical win works in the first column when not enough tokens to win.

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 4	<table><tr><td></td><td>0</td><td>…</td><td>7</td><td>8</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td>x</td><td></td><td></td><td></td></tr><tr><td>7</td><td></td><td>X</td><td></td><td></td><td></td></tr><tr><td>8</td><td></td><td>X</td><td></td><td></td><td></td></tr><tr><td>9</td><td></td><td>X</td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks if vertical win works in the second column when it hits the number to win

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 4	<table><tr><td></td><td>0</td><td>…</td><td>4</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td>X</td><td></td><td></td></tr><tr><td>7</td><td></td><td></td><td>X</td><td></td><td></td></tr><tr><td>8</td><td></td><td></td><td>X</td><td></td><td></td></tr><tr><td>9</td><td></td><td></td><td>x</td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks if board matches requirements, vertical win works in the second column when it hits the number to win

Input:	Output:	Reason:
Row = 5 Col = 5 Win = 3	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td>0</td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td>0</td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td>0</td><td></td><td></td></tr><tr><td>4</td><td></td><td></td><td>0</td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks if board matches requirements, vertical win works in the third column when it hits the number to win

- CheckDiagWin

Input:	Output:	Reason:
Row = 6 Col = 6 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td>o</td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td>o</td><td>X</td><td></td></tr><tr><td>4</td><td>o</td><td></td><td>o</td><td>X</td><td>O</td><td></td></tr><tr><td>5</td><td>X</td><td>o</td><td>X</td><td>x</td><td>X</td><td></td></tr></table>	This test case is unique and distinct because it checks top diagonal right.

Input:	Output:	Reason:
Row = 6 Col = 6 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td>O</td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td>o</td><td>X</td><td></td></tr><tr><td>4</td><td>o</td><td></td><td>o</td><td>X</td><td>O</td><td></td></tr><tr><td>5</td><td>x</td><td>o</td><td>X</td><td>x</td><td>X</td><td></td></tr></table>	This test case is unique and distinct because it checks bottom diagonal right.

Input:	Output:	Reason:
Row = 6 Col = 6 Win = 4		This test case is unique and distinct because it checks middle slanted right

	0	1	2	3	4	5
0						
1						
2					o	
3				o	X	
4	o		o	X	O	
5	x	o	X	X	X	

Input:	Output:	Reason:
Row = 5 Col = 5 Win = 4		This test case is unique and distinct because it checks top diagonal slanted left.

	0	1	2	3	4
0					
1		x	x		
2		O	X		
3		O	o	X	
4	x	o	X	o	X

Input:	Output:	Reason:
Row = 5 Col = 5 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td>x</td><td>x</td><td></td><td></td></tr><tr><td>2</td><td></td><td>O</td><td>X</td><td></td><td></td></tr><tr><td>3</td><td></td><td>O</td><td>o</td><td>X</td><td></td></tr><tr><td>4</td><td>x</td><td>o</td><td>X</td><td>o</td><td>X</td></tr></table>	This test case is unique and distinct because it checks bottom diagonal slanted left.

Input:	Output:	Reason:
Row = 5 Col = 5 Win = 4	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td>x</td><td>x</td><td></td><td></td></tr><tr><td>2</td><td></td><td>O</td><td>X</td><td></td><td></td></tr><tr><td>3</td><td></td><td>O</td><td>o</td><td>X</td><td></td></tr><tr><td>4</td><td>x</td><td>o</td><td>X</td><td>o</td><td>X</td></tr></table>	This test case is unique and distinct because it checks middle diagonal slanted left.

Input:	Output:	Reason:
Row = 3 Col = 4 Win = 3	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td>x</td><td></td><td>o</td><td>x</td></tr><tr><td>1</td><td>O</td><td>x</td><td>o</td><td>x</td></tr><tr><td>2</td><td>x</td><td>o</td><td>X</td><td>O</td></tr></table>	This test case is unique and distinct because it checks if the last token placed is a diagonal win

Input:	Output:	Reason:
Row = 3 Col = 4 Win = 3	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td>x</td><td>o</td><td>o</td><td>x</td></tr><tr><td>1</td><td>O</td><td>x</td><td>o</td><td>x</td></tr><tr><td>2</td><td>x</td><td>o</td><td>X</td><td>O</td></tr></table>	This test case is unique and distinct because it checks if the last token placed is a diagonal win when the board is full

- CheckTie

Input:	Output:	Reason:
Row = 3 Col = 4 Win = 3	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it tests for a tie on an empty board.

Input:	Output:	Reason:
Row = 3 Col = 3 Win = 3	<table><tr><td></td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>x</td><td>o</td><td>x</td></tr><tr><td>1</td><td>o</td><td>x</td><td>O</td></tr><tr><td>2</td><td>x</td><td>o</td><td>X</td></tr></table>	This test case is unique and distinct because it tests for a tie on a full board.

Input:

Row = 10
Col = 10
Win = 4

Output:

	0	1	2	3	…	9
0						
…						
6						
7						
8		o	o	o	o	
9		x	x	x	b	

Reason:

This test case is unique and distinct because it tests for a tie when there is a win

Input:

Row = 10
Col = 10
Win = 10

Output:

	0	1	2	3	4	5	6	7	8	9
0	x	o	p	g	n	a	y	r	q	X
1	x	o	p	g	n	a	y	r	q	X
2	x	o	p	g	n	a	y	r	q	X
3	x	o	p	g	n	a	y	r	q	X
4	x	o	p	g	n	a	y	r	q	X
5	x	o	p	g	n	a	y	r	q	X
6	x	o	p	g	n	a	y	r	q	X
7	x	o	p	g	n	a	y	r	q	X
8	x	o	p	g	n	a	y	r	q	X
9	x	o	p	g	n	a	y	r	q	X

Reason:

This test case is unique and distinct because it tests for a tie when the board is full

- isPlayerAtPos

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it tests for a character at a position on an empty board.

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>x</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it tests for a character at the bottom left of the board.

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td>x</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it tests for when two characters are in the same column

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td>o</td><td>o</td><td>o</td><td>o</td><td>o</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td>o</td><td>o</td><td>r</td><td>o</td><td>O</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it tests for a character in the middle of other characters

<table>
<tr>
<td>Input:

Row = 10
Col = 10
Win = 10</td>
<td>Output:

<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td>
<td>Reason:

This test case is unique and distinct because it tests for a character in that is in a different column</td>
</tr>
</table>

- WhatsAtPos

<table>
<tr>
<td>Input:

Row = 10
Col = 10
Win = 10</td>
<td>Output:

<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td>
<td>Reason:

This test case is unique and distinct because it checks to see what is at a position in an empty board.</td>
</tr>
</table>

<table>
<tr>
<td>Input:

Row = 10
Col = 10
Win = 10</td>
<td>Output:

<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td>
<td>Reason:

This test case is unique and distinct because it checks to see what is at a position in a board with only one character in it.</td>
</tr>
</table>

<table>
<tr>
<td>Input:

Row = 10
Col = 10
Win = 10</td>
<td>Output:

<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>x</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td>
<td>Reason:

This test case is unique and distinct because it checks to see what is at a the bottom row in a board.</td>
</tr>
</table>

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td>o</td><td>o</td><td>o</td><td>o</td><td>o</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td>o</td><td>o</td><td>r</td><td>o</td><td>O</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it tests for what's in the middle of other characters.

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks to see what is at a position in a board next to a character

- placeToken

Input:	Output:	Reason:
Row = 5 Col = 5 Win = 3	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>4</td><td>x</td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks to see if placetoken placed the character in the correct row on an empty board.

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>x</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks to see if placetoken placed the character in the correct spot when the column is partially full

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td>x</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td>O</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td>O</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td>O</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td>O</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>O</td><td></td><td></td><td></td><td></td><td></td></tr></table>	This test case is unique and distinct because it checks to see if placetoken placed the character in the correct spot when the column is full

Input:	Output:	Reason:
Row = 10 Col = 10 Win = 10	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>…</td><td>9</td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>9</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>o</td></tr></table>	This test case is unique and distinct because it checks to see if placetoken placed the character in the correct spot when the row is full

<table>
<tr>
<td>Input:

Row = 5
Col = 5
Win = 5</td>
<td>Output:

<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td>0</td><td>o</td><td>x</td><td>o</td><td>x</td><td>o</td></tr>
<tr><td>1</td><td>o</td><td>x</td><td>o</td><td>x</td><td>O</td></tr>
<tr><td>2</td><td>x</td><td>o</td><td>x</td><td>o</td><td>X</td></tr>
<tr><td>3</td><td>o</td><td>x</td><td>o</td><td>x</td><td>O</td></tr>
<tr><td>4</td><td>x</td><td>o</td><td>x</td><td>o</td><td>X</td></tr>
</table>
</td>
<td>Reason:

This test case is unique and distinct because it checks to see if placetoken placed the character in the correct spot when the board is full</td>
</tr>
</table>