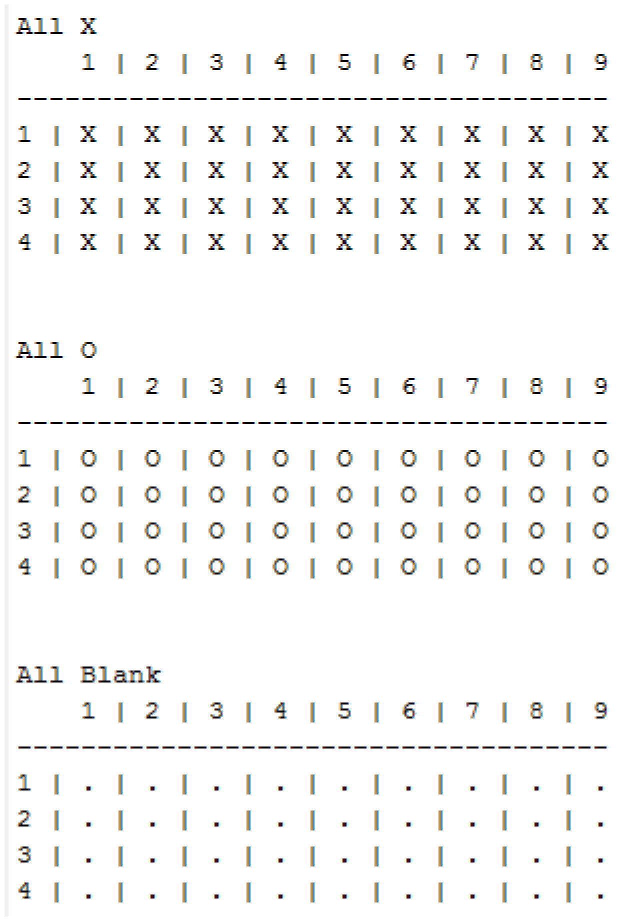
MATLAB Assignment Report

Testing Individual Functions

Testing *printBoard.m*

This function neatly prints the current status of the board. To test *printBoard.m*, the script *testPrintBoard.m* was used. The test script printed all the game characters separately, with vertical bars used to demarcate squares. The result displayed in the command window was as shown in Figure 1.



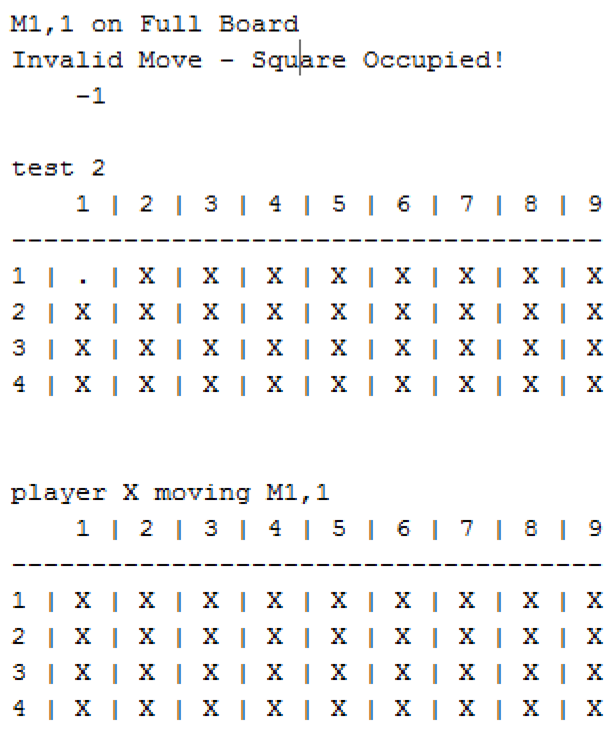
```
All X
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | X | X | X | X | X | X | X | X | X
2 | X | X | X | X | X | X | X | X | X
3 | X | X | X | X | X | X | X | X | X
4 | X | X | X | X | X | X | X | X | X


All O
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | O | O | O | O | O | O | O | O | O
2 | O | O | O | O | O | O | O | O | O
3 | O | O | O | O | O | O | O | O | O
4 | O | O | O | O | O | O | O | O | O


All Blank
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | . | . | . | . | . | . | . | . | .
4 | . | . | . | . | . | . | . | . | .
```

Figure 1: Printing different game characters using printBoard.m

Testing *playHumanMove.m*

This function registers the move made by the human player on the board. The inputs are the status of the board before the move, the move made and the character used by the human player. The output is the updated status of the board after the move. The move string must be in the format prescribed in the function documentation.

To test this function, the script *testplayHumanMove.m* was used. The script fills up the board with move characters then attempts to make a move. Such is not allowed because the board is full. A space is then freed up and the move attempted again. With the new status of the board the move becomes successful. The results are shown in Figure 2.



```
M1,1 on Full Board
Invalid Move - Square Occupied!
    -1

test 2
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | X | X | X | X | X | X | X | X
2 | X | X | X | X | X | X | X | X | X
3 | X | X | X | X | X | X | X | X | X
4 | X | X | X | X | X | X | X | X | X


player X moving M1,1
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | X | X | X | X | X | X | X | X | X
2 | X | X | X | X | X | X | X | X | X
3 | X | X | X | X | X | X | X | X | X
4 | X | X | X | X | X | X | X | X | X
```

Figure 2: *playHumanMove.m* test results

Testing *computerMove.m*

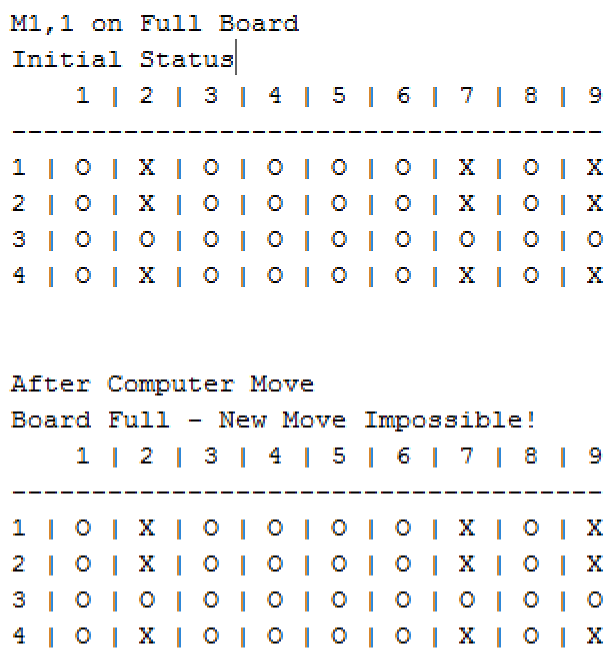
The strategy used in computer move involved checking the sequence of opponents pieces with a prospect of giving the win to the opponent in one or two moves to follow. If there is a threat posed by the opponent in the next move, the computer moves to place its piece so that the sequence is thwarted. If there are multiple such sequences, then the computer can only stop one. The highest threat is posed by three opponent pieces so arranged that adding another piece to the three pieces forms a diagonal, horizontal or vertical row. If any such patterns are detected, the computer places a piece to try and stop the formation of a row. If there are multiple was of completing the row then the computer loses.

Another threat is posed by two pieces, adjacent to each other horizontally, and with spaces to the right and to the left still open. If during the present move the computer fails to make a move that tries to stop the sequence, the opposing player adds the third piece in the sequence with the next move and even if the computer attempt to stop the completion of the sequence, the opposing player still gets an opportunity to win the game because there are more than a single way of completing the row.

The diagonal, horizontal and vertical sequence searches are implemented in aRow.m.

Testing was done using the test computerMove.m script. The first test attempts to make a computer move in a full board. The results are as shown in Figure 3. The status of the board remains as was.

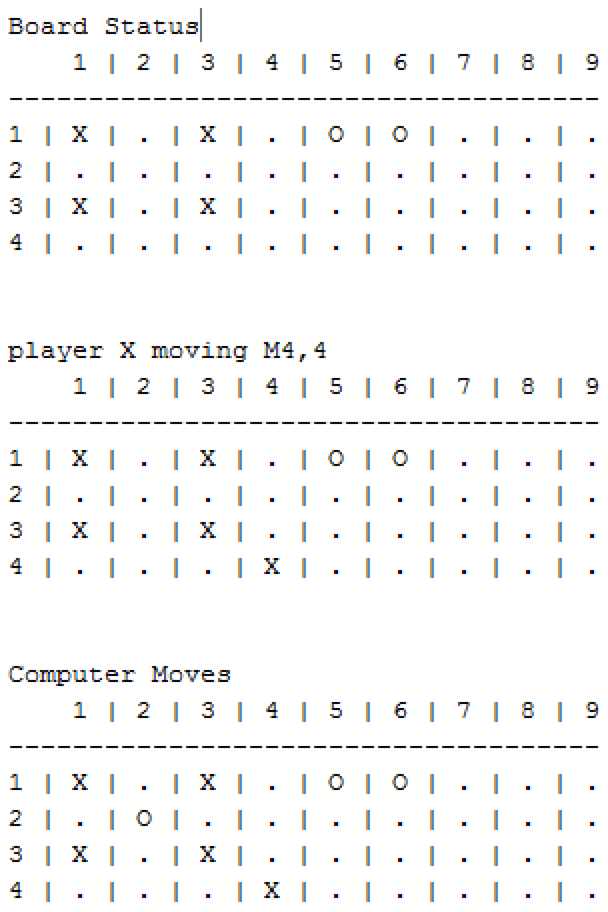
The second test frees up space on the board, presents the board in a given state, lets the opponent make a threatening move and the computer moves to thwart the opponents impending win. The results were as shown in Figure 4.



```
M1,1 on Full Board
Initial Status
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | O | X | O | O | O | O | X | O | X
2 | O | X | O | O | O | O | X | O | X
3 | O | O | O | O | O | O | O | O | O
4 | O | X | O | O | O | O | X | O | X


After Computer Move
Board Full - New Move Impossible!
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | O | X | O | O | O | O | X | O | X
2 | O | X | O | O | O | O | X | O | X
3 | O | O | O | O | O | O | O | O | O
4 | O | X | O | O | O | O | X | O | X
```

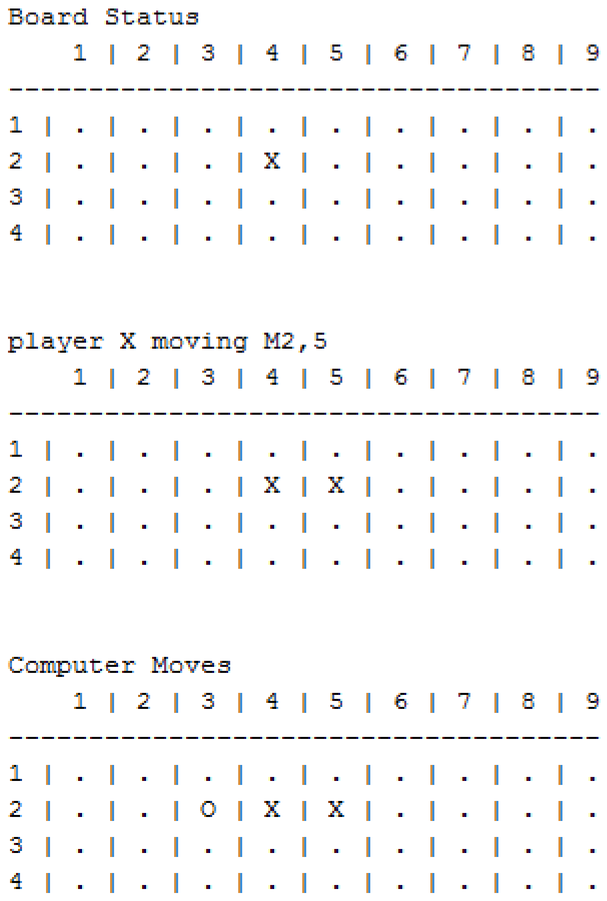
Figure 3: Attempting a move in a full board

In another test, the opponent's two pieces are adjacent to each other horizontally with spaces to the left and the right. The computer moves to break any possible addition of a third peace to prevent the opponent from having more than two ways of completing the row. The result is shown in Figure 5. If the computer fails to make this move, the opponent adds a third piece as shown in Figure 6 and opens up two possible completions of the row, which the computer cannot thwart.



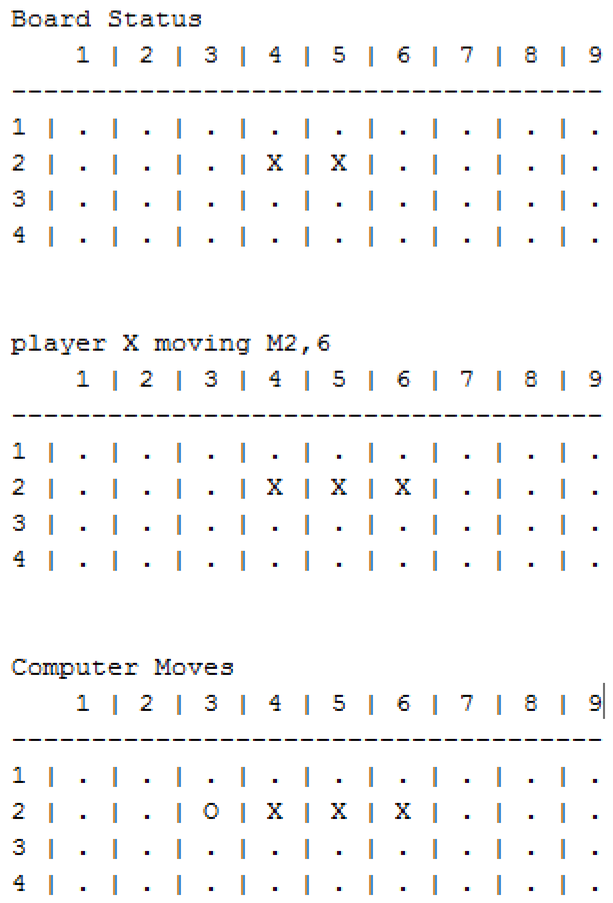
```
Board Status
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | X | . | X | . | O | O | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | X | . | X | . | . | . | . | . | .
4 | . | . | . | . | . | . | . | . | .


player X moving M4,4
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | X | . | X | . | O | O | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | X | . | X | . | . | . | . | . | .
4 | . | . | . | X | . | . | . | . | .


Computer Moves
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | X | . | X | . | O | O | . | . | .
2 | . | O | . | . | . | . | . | . | .
3 | X | . | X | . | . | . | . | . | .
4 | . | . | . | X | . | . | . | . | .
```

Figure 4: Computer move to thwart an impending opponent win



```
Board Status
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | X | . | . | . | . | .
3 | . | . | . | . | . | . | . | . | .
4 | . | . | . | . | . | . | . | . | .


player X moving M2,5
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | X | X | . | . | . | .
3 | . | . | . | . | . | . | . | . | .
4 | . | . | . | . | . | . | . | . | .


Computer Moves
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | O | X | X | . | . | . | .
3 | . | . | . | . | . | . | . | . | .
4 | . | . | . | . | . | . | . | . | .
```

Figure 5: Computer thwarts the threat posed by two horizontally adjacent pieces with open spaces to the left and right



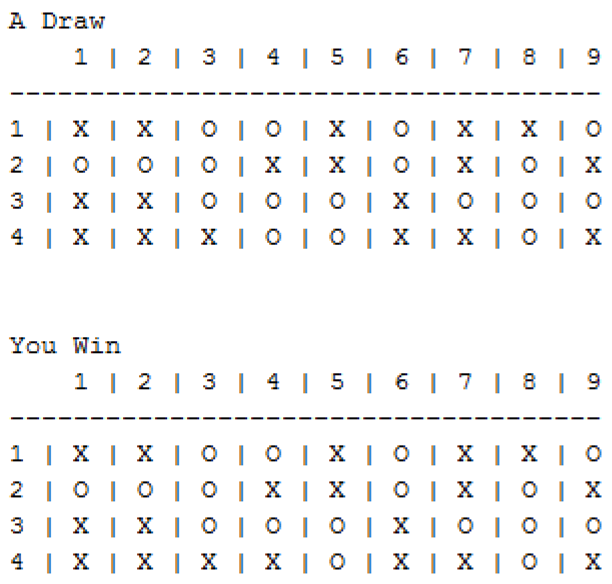
```
Board Status
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | X | X | . | . | . | .
3 | . | . | . | . | . | . | . | . | .
4 | . | . | . | . | . | . | . | . | .


player X moving M2,6
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | X | X | X | . | . | .
3 | . | . | . | . | . | . | . | . | .
4 | . | . | . | . | . | . | . | . | .


Computer Moves
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | O | X | X | X | . | . | .
3 | . | . | . | . | . | . | . | . | .
4 | . | . | . | . | . | . | . | . | .
```

Figure 6: Two possible endings to a sequence leading to a failure to thwart the impending opponent win

Testing *checkVictory.m*

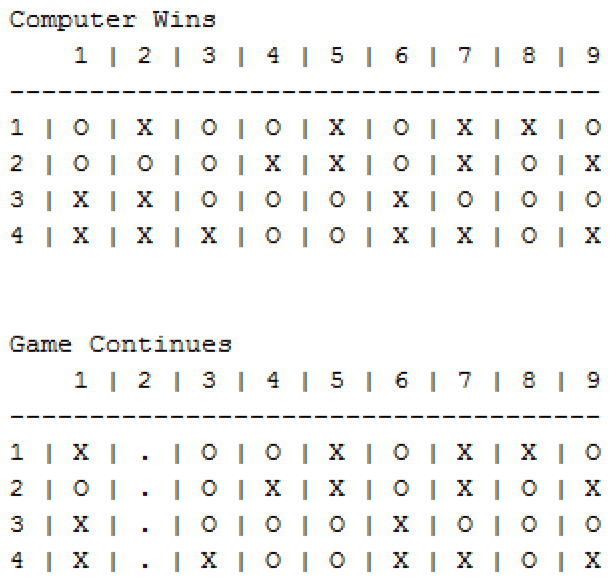
This function uses the board status after a mover by a player, and the piece used by the human player, to determine whether or not the game produces a winner, and if it does, who that winner is, to determine whether or not the game is played to a draw and equally critically, to determine whether or not the game is still in progress. The function aRow.m is used to conduct various sequence searches.

The test script *testcheckVictory.m* was used to test the function for each of the four outputs. The results were as shown in Figure 7 and Figure 8



```
A Draw
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | X | X | O | O | X | O | X | X | O
2 | O | O | O | X | X | O | X | O | X
3 | X | X | O | O | O | X | O | O | O
4 | X | X | X | O | O | X | X | O | X


You Win
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | X | X | O | O | X | O | X | X | O
2 | O | O | O | X | X | O | X | O | X
3 | X | X | O | O | O | X | O | O | O
4 | X | X | X | X | O | X | X | O | X
```

Figure 7: A draw and human player win

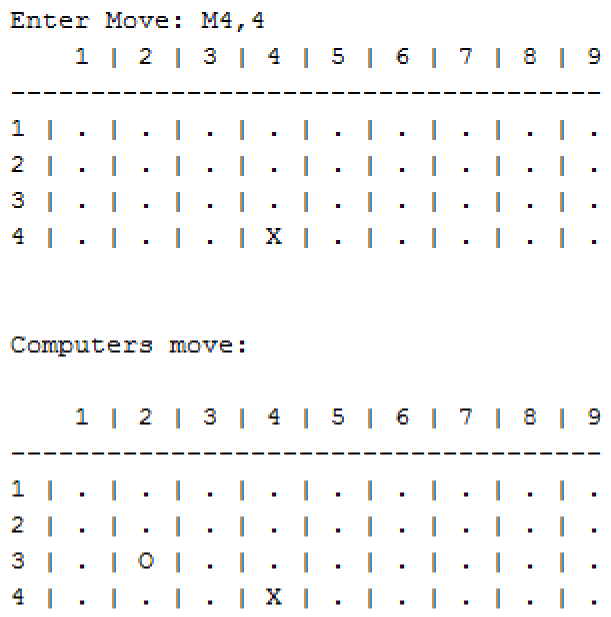


```
Computer Wins
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | O | X | O | O | X | O | X | X | O
2 | O | O | O | X | X | O | X | O | X
3 | X | X | O | O | O | X | O | O | O
4 | X | X | X | O | O | X | X | O | X


Game Continues
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | X | . | O | O | X | O | X | X | O
2 | O | . | O | X | X | O | X | O | X
3 | X | . | O | O | O | X | O | O | O
4 | X | . | X | O | O | X | X | O | X
```

Figure 8: A computer win and a case for continuing game

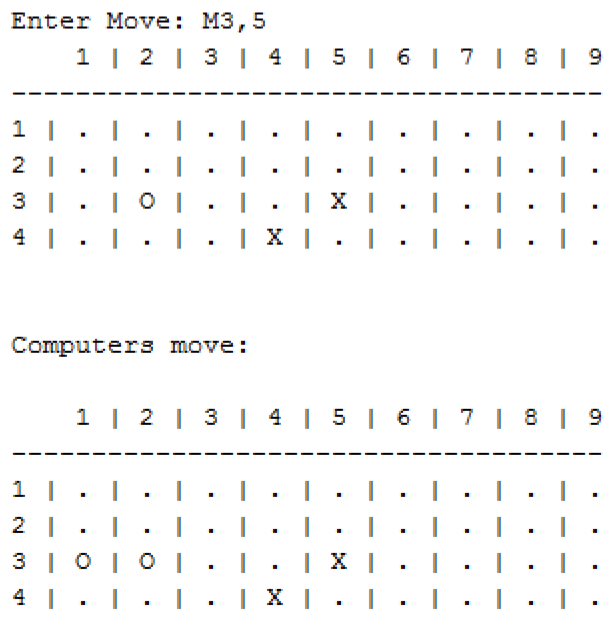
Testing *nac.m*

This script was modified to give the human player the opportunity to select the correct character, that is, either upper case ‘O’ or uppercase ‘X’. The game sequence always gives the starting opportunity to the human player. The game continues provided the board is not full and there is no winner. A sample game sequence was as shown in Figure 9 to Figure 12.



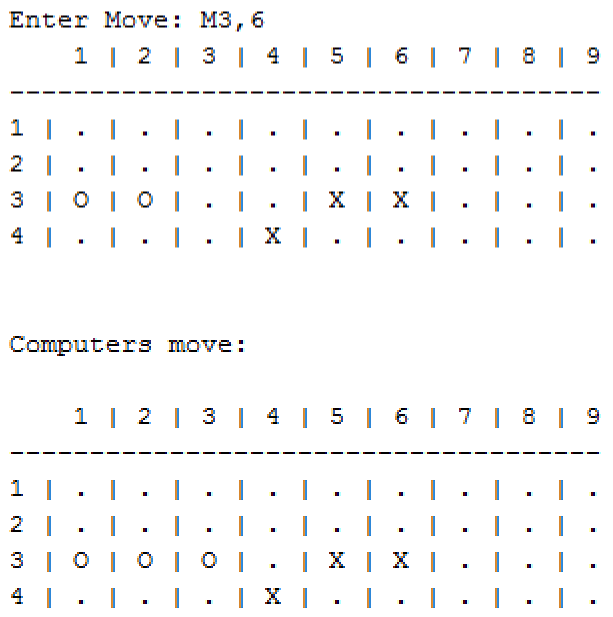
```
Enter Move: M4,4
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | . | . | . | . | . | . | . | . | .
4 | . | . | . | X | . | . | . | . | .


Computers move:

    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | . | O | . | . | . | . | . | . | .
4 | . | . | . | X | . | . | . | . | .
```

Figure 9: First move



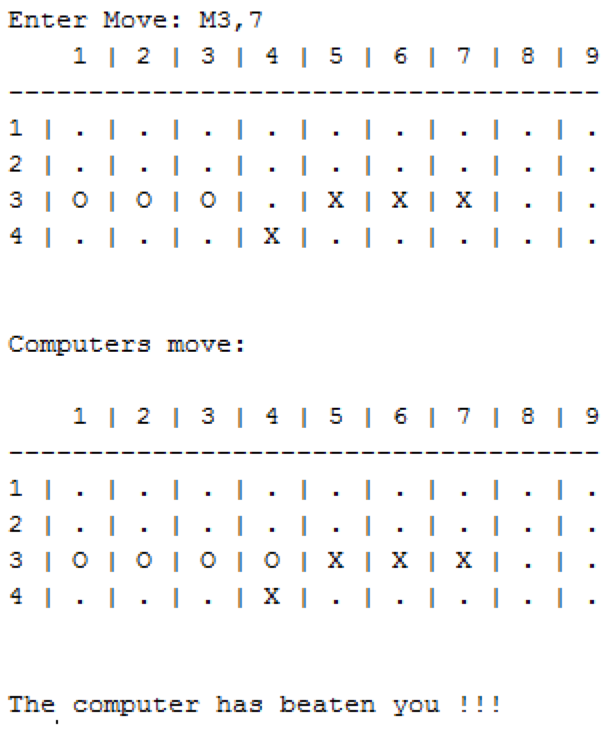
```
Enter Move: M3,5
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | . | O | . | . | X | . | . | . | .
4 | . | . | . | X | . | . | . | . | .


Computers move:

    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | O | O | . | . | X | . | . | . | .
4 | . | . | . | X | . | . | . | . | .
```

Figure 10: Second move



```
Enter Move: M3,6
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | O | O | . | . | X | X | . | . | .
4 | . | . | . | X | . | . | . | . | .


Computers move:

    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | O | O | O | . | X | X | . | . | .
4 | . | . | . | X | . | . | . | . | .
```

Figure 11: Third Move



```
Enter Move: M3,7
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | O | O | O | . | X | X | X | . | .
4 | . | . | . | X | . | . | . | . | .


Computers move:

    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
-------------------------------------
1 | . | . | . | . | . | . | . | . | .
2 | . | . | . | . | . | . | . | . | .
3 | O | O | O | O | X | X | X | . | .
4 | . | . | . | X | . | . | . | . | .


The computer has beaten you !!!
```

Figure 12: Fourth Move