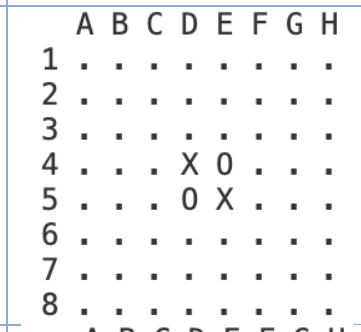
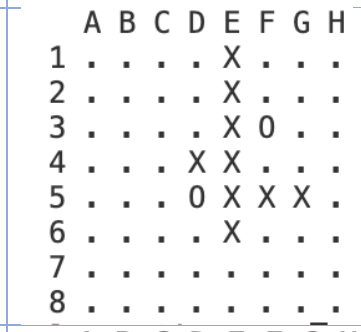
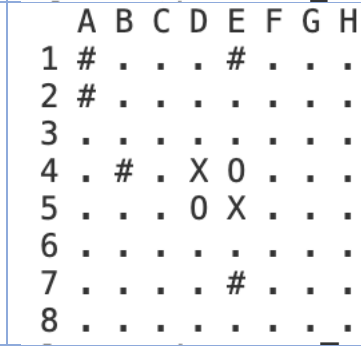
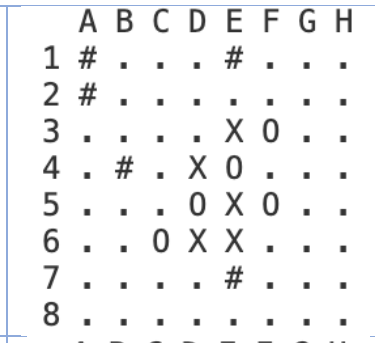
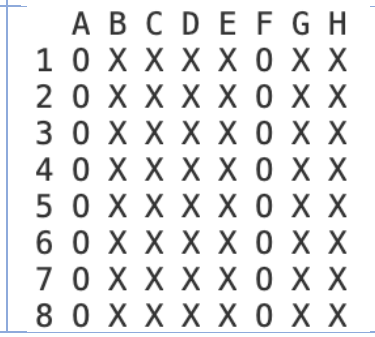
CSCI1120 Marking Scheme
Assignment 4 Reversi

2. Test cases

2.1 Function Unit Test for has_empty_cells

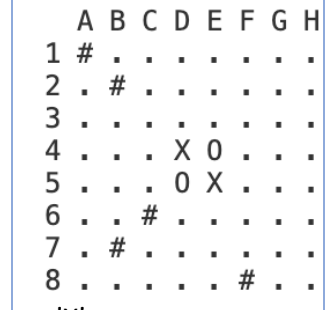
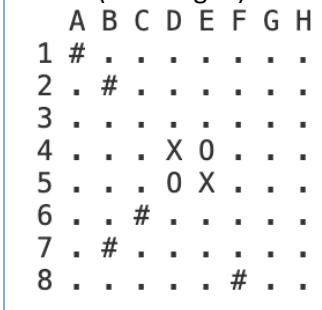
Test the has_empty_cells function with different conditions to verify the correctness of return value.

Case	Inputs	Outputs	Description
1		true	Test board's initial state as basic case
2		true	Test board with some discs placed and no blocks and there exist some empty cells
3		true	Test board with blocks and initial discs , and there exist some empty cells

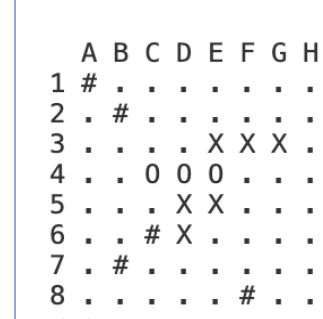
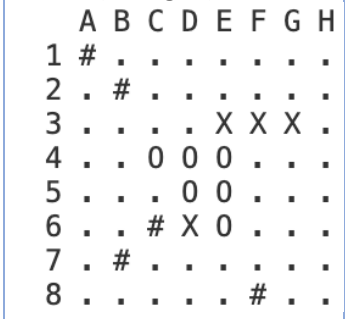
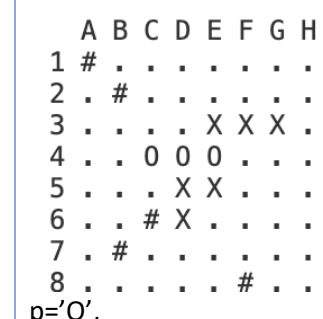
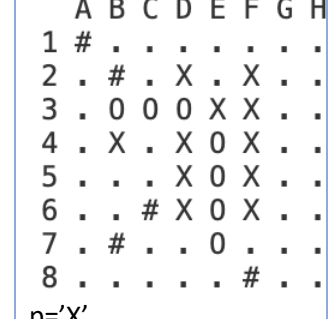
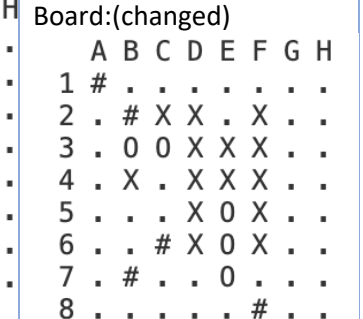
4		true	Test board with blocks and new discs placed, and there exist some empty cells
5		false	Test a board full with discs and no empty cells. This is not a real case but to test this function.

2.2 Function Unit Test for valid_move

Test the valid_move function with different inputs to verify the functionality. If the flip is set to true, we need to compare the content of the board array.

Case	Inputs	Outputs	Description
1	Board:  p='X', y=-1,x=0, (A0) flip=false	Output:False Board:(unchanged) 	Test the invalid move which is out of board (A0 in this case)
2	Board:	Output:False Board:(unchanged)	Test the invalid move which is to be placed on the existing disc (E5 in this case)

	A B C D E F G H 1 # . . . . 2 . # . . . . . . 3 . . . . 4 . . . X O . . . 5 . . . O X . . . 6 . . # . . . . . 7 . # . . . . . . 8 . . . . . # . . p='X', y=4,x=4, (E5) flip=false	A B C D E F G H 1 # . . . . 2 . # . . . . . . 3 . . . . 4 . . . X O . . . 5 . . . O X . . . 6 . . # . . . . . 7 . # . . . . . . 8 . . . . . # . .	
3	Board: A B C D E F G H 1 # . . . . 2 . # . . . . . . 3 . . . . 4 . . . X O . . . 5 . . . O X . . . 6 . . # . . . . . 7 . # . . . . . . 8 . . . . . # . . p='X', y=1,x=1, (B2) flip=false	Output:False Board:(unchanged) A B C D E F G H 1 # . . . . 2 . # . . . . . . 3 . . . . 4 . . . X O . . . 5 . . . O X . . . 6 . . # . . . . . 7 . # . . . . . . 8 . . . . . # . .	Test the invalid move which is to be placed on the existing block (B2 in this case)
4	Board: A B C D E F G H 1 # . . . . 2 . # . . . . . . 3 . . . . 4 . . . X O . . . 5 . . . O X . . . 6 . . # . . . . . 7 . # . . . . . . 8 . . . . . # . . p='X', y=5,x=5, (F6) flip=false	Output:False Board:(unchanged) A B C D E F G H 1 # . . . . 2 . # . . . . . . 3 . . . . 4 . . . X O . . . 5 . . . O X . . . 6 . . # . . . . . 7 . # . . . . . . 8 . . . . . # . .	Test the invalid move which is placed on an empty cell. (F6 in this case)
5	Board: A B C D E F G 1 # . . . . . . 2 . # . . . . . 3 . . . . X O . 4 . . . X O . . 5 . . . O X . . 6 . . # . . . . 7 . # . . . . . 8 . . . . . # .	Output:True Board:(change) A B C D E F G H 1 # . . . . 2 . # . . . . . . 3 . . . . X O . . 4 . . . X O . . . 5 . . . X X . . . 6 . . # X . . . . 7 . # . . . . . . 8 . . . . . # . .	Test a valid move and change the board's content as the flip is set to true.

	p='X', y=5,x=3, (D6) flip=true		
6	Board:  p='O', y=5,x=4, (E6) flip=true	Output:True Board:(changed) 	Test a valid move and do the flipping in two directions.
7	Board:  p='O', y=5,x=4, (E6) flip=false	Output:True Board:(unchanged)	Test a valid move and no flipping.
8	Board:  p='X', y=1,x=2, (C2) flip=true	Output:True Board:(changed) 	Test a valid move which needs to search multiple steps to find a path.
9	Board:	Output:True Board:(changed)	A relatively hard case.

	A B C D E F G H 1 # . . . . 2 . # X X . X . . 3 . 0 0 X X X . . 4 . X . X X X 0 . 5 . X X X X 0 . . 6 . . # X 0 X . . 7 . # X . 0 . . . 8 . . . . . # . . p='O', y=4,x=0, (A5) flip=true	A B C D E F G H 1 # . . . . 2 . # X X . X . . 3 . 0 0 X X X . . 4 . 0 . X X X 0 . 5 0 0 0 0 0 0 . . 6 . . # X 0 X . . 7 . # X . 0 . . . 8 . . . . . # . .	
10	Board: A B C D E F G H 1 # . . . . 2 . # X X . X . . 3 . 0 0 X X X . . 4 . X . X X X 0 . 5 . X X X X 0 . . 6 . . # X 0 X . . 7 . # X . 0 . . . 8 . . . . . # . . p='O', y=5,x=1, (B6) flip=true	Output:True Board:(changed) A B C D E F G H 1 # . . . . 2 . # X X . X . . 3 . 0 0 X X X . . 4 . 0 . X X X 0 . 5 . 0 X X X 0 . . 6 . 0 # X 0 X . . 7 . # X . 0 . . . 8 . . . . . # . .	A hard case with blocks next to the move position.
11	Board: A B C D E F G H 1 . . . . 2 . . . . 3 . . # # # # . . 4 . . # X 0 # . . 5 . . # 0 X # . . 6 . . # # . # . . 7 . . . . 8 . . . . p='X', y=4,x=1, (B5) flip=false	Output:False Board:(unchanged)	A case where the block breaks the only possible path.

2.3 Function Unit Test for has_valid_move

Test the has_valid_move function with different conditions to verify the correctness of return value.

Case	Inputs	Outputs (network value)	Description

1	Board: A B C D E F G H 1 . . . . 2 . . . . 3 . . # # # # . . 4 . . # X O # . . 5 . . # O X # . . 6 . . # # . # . . 7 . . . . 8 . . . . p='X'	false	The blocks surround the placed discs
2	Board: A B C D E F G H 1 # . . . . 2 . # X X . X . . 3 . O O X X X . . 4 . X . X X X . . 5 . . . X O X . . 6 . . # X O X . . 7 . # . . O . . . 8 . . . . . # . . p='O'	True	A relative hard case
3	Board: A B C D E F G H 1 # . . . . 2 . # . . . . . . 3 . . . . 4 . . . X O . . . 5 . . . O X . . . 6 . . # . . . . . 7 . # . . . . . . 8 . . . . . # . . P='X'	True	A simple case
4	Board: A B C D E F G H 1 . . . . 2 . . . . 3 . . . . # # . . 4 . . . X O # . . 5 . . # O X . . . 6 . . . # . . . . 7 . . . . 8 . . . . p='x'	False	The potential valid moves for 'x' are all occupied by blocks.

2.4 Gameflow Test

Here we use 3 test flows to test the implementation of gameflow.

2.4.1 Flow1

In flow1, we will set 21 blocks in a 8*8 board. And x will win the game.

Step	Inputs	Output	Description
1	33	Too many blocks!	Input an Invalid block number.
2	21		Input a valid block number
3	A0	Invalid position!	Invalid block position out of board
4	A2		Valid block position
5	D4	Invalid position!	Invalid block position on existing disc
6	B4		Valid block position
7	D3		Valid block position
8	G6		Valid block position
9	B7		Valid block position
10	F7		Valid block position
11	G2		Valid block position
12	F6		Valid block position
13	C4		Valid block position
14	C5		Valid block position
15	C5	Invalid position!	Invalid block position on existing block
16	C3		Valid block position
17	C6		Valid block position
18	C7		Valid block position
19	D7		Valid block position
20	E7		Valid block position
21	D2		Valid block position
22	E2		Valid block position
23	F2		Valid block position
24	G3		Valid block position
25	G4		Valid block position
26	G5		Valid block position

After block initialization:

```
  A B C D E F G H
1 . . . . . . . .
2 # . . # # # # .
3 . . # # . . # .
4 . # # X 0 . # .
5 . . # 0 X . # .
6 . . # . . # # .
7 . # # # # # . .
8 . . . . . . . .
```

(If any element in the board is wrong, stop the evaluation.)

27	E3		Valid move for X
28	F3		Valid move for O
29	D6		Valid move for X
30	E6		Valid move for O
31	H3	Invalid move!	Invalid move for X
32	F4		Valid move for X
33	F5		Valid move for O
34		Player X has no valid moves! Pass!	No valid move for X
35		Player O has no valid moves! Pass!	No valid move for Y
36		Player X wins!	Game End

Final state:

```
  A B C D E F G H
1 . . . . . . . .
2 # . . # # # # .
3 . . # # X O # .
4 . # # X X O # .
5 . . # X X O # .
6 . . # X O # # .
7 . # # # # # . .
8 . . . . . . . .
```

(If any element in the board is wrong, it will cause a deduction of 4'.)

2.4.2 Flow2

In flow2, we will initialize a 6*6 board with no block. And O will win the game.

Step	Inputs	Output	Description
1	0		input block number

After block initialization:

```
  A B C D E F
1 . . . . . .
2 . . . . . .
3 . . X O . .
4 . . O X . .
5 . . . . . .
6 . . . . . .
```

(If any element in the board is wrong, stop the evaluation.)

2	D2		Valid move for X
3	B4	Invalid move!	Invalid move for O
4	E4		Valid move for O
5	F9	Invalid move!	Invalid move for X
6	B5		Valid move for X
7	B4		Valid move for O
8	A5		Valid move for X
9	D1		Valid move for O
10	A6	Invalid move!	Valid move for X
11	F4		Valid move for X

12	A4	Invalid move!	Valid move for O
13	D5		Valid move for O
14	E5		Valid move for X
15	F5		Valid move for O
16	D6		Valid move for X
17	C5		Valid move for O
18	B6		Valid move for X
19	A4		Valid move for O
20	E3		Valid move for X
21	E2		Valid move for O
22	F3		Valid move for X
23	B2		Valid move for O
24	F2	Invalid move!	Invalid move for X
25	F6		Valid move for X
26	A6		Valid move for O
27	C6		Valid move for X
28	E6		Valid move for O
29	A3		Valid move for X
30	B3		Valid move for O
31	E1		Valid move for X
32	F2		Valid move for O
33	F1		Valid move for X
34	A2		Valid move for O
35	G1	Invalid move!	Invalid move for X
36	C1		Valid move for X
37		Player O has no valid moves! Pass!	
38	C2		Valid move for X
39		Player O has no valid moves! Pass!	
40	B1	Invalid move!	Invalid move for X
41	A1		Valid move for X
42	B1		Valid move for O
		Player X wins!	The game ends.

Final State:

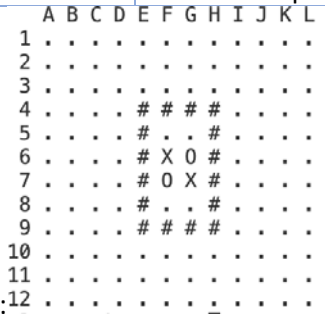
```
Game over:
  A B C D E F
1 X O X X X X
2 O O X X X X
3 O O X X X X
4 O O X O X X
5 O O X O O X
6 O O O O O X
Player X wins!
```

(If any element in the board is wrong, it will cause a deduction of 4'.)

2.4.3 Flow3

In flow3, we will set 16 blocks in a 12*12 board. It's a draw game.

Step	Inputs	Output	Description
1	16		Input the block number
2	A13	Invalid position!	Invalid position(out of range)
3	E5		Valid block position
4	E4		Valid block position
5	F4		Valid block position
6	G4		Valid block position
7	H4		Valid block position
8	H5		Valid block position
9	M1	Invalid position!	Invalid position(out of range)
10	H6		Valid block position
11	H7		Valid block position
12	H8		Valid block position
13	H9		Valid block position
14	G9		Valid block position
15	F9		Valid block position
16	E9		Valid block position
17	E8		Valid block position
18	E7		Valid block position
19	F6	Invalid position!	Invalid position(on existing disc)
20	E6		Valid block position
After block initialization: **(If any element in the board is wrong, stop the evaluation.)**			
21	G5		Valid move for X
22	I7	Invalid move!	Invalid move for O
23	F5		Valid move for O
24		Player X has no valid moves! Pass!	No valid move for X
25		Player O has no valid moves! Pass!	No valid move for O
26		Draw game!	game over



Final State:

```
  A B C D E F G H I J K L
1 . . . . . . . . . . . .
2 . . . . . . . . . . . .
3 . . . . . . . . . . . .
4 . . . . # # # # . . . .
5 . . . . # O X # . . . .
6 . . . . # O X # . . . .
7 . . . . # O X # . . . .
8 . . . . # . . # . . . .
9 . . . . # # # # . . . .
10 . . . . . . . . . . . .
11 . . . . . . . . . . . .
12 . . . . . . . . . . . .
```

(If any element in the board is wrong, it will cause a deduction of 3'.)