reversal_test

August 24, 2020

1 Testing Reversal

Here's the list of functions for this reversal game:

1. `create_board()`: Initialises an empty board (a numpy array) with 4 pieces in the centre.
2. `can_play(board, piece)`: Determines whether or not the user can place a piece, by performing `is_reversible()` on all vacant spots on the board. (In some cases a player may be inhibited to do so since there is no available spot where a reversal can occur.)
3. `is_vacant(board, row, col, piece)`: Determines whether a position on a board is empty.
4. `is_reversible(board, row, col, piece)`: Determines whether or not placing a piece at the given location would lead to reversals.
5. `drop_piece(board, row, col, piece)`: Performed after `is_reversible()`. The piece is placed at the specified location and performs all reversals required. Also returns an integer, which is the number of reversals.
6. `print_board(board, flip_num)`: Prints the board (both actual terminal output and a 'heat map' for better visibility), the total number of pieces on the board, the number of pieces for each player, and the number of flips (reversals) performed.
7. `def is_end_game(board)`: Boolean true if all entries on the board are non-zero (so basically the board is full).
8. `def insert(row, col, piece)`: A function specifically designed for the jupyter workbook, which performs `drop_piece()` and `print_board()`. Note that the notebook does not actually perform verification steps such as `can_play()`, `is_vacant()`, `is_reversible()`.

1.1 List of functions and packages used.

[1]:
```python
import numpy as np
import matplotlib.pylab as plt

# Global variables
DIM = 8

############## INITIALISE VARIABLES ######################
game_over = False
```

```
turn = 1
error = False
p1_score = 0
p2_score = 0

################# FUNCTIONS #######################

# Function to Initiate Board
def create_board():
    board = np.zeros((DIM,DIM))
    board[3][3] = 1
    board[4][4] = 1
    board[3][4] = 2
    board[4][3] = 2
    return board

# Function to check if player can place piece.
def can_play(board, piece):
    for r in range(DIM):
        for c in range(DIM):
            if is_vacant(board, r, c, piece):
                if is_reversible(board, r, c, piece):
                    return True
    return False

# Function to check if location is vacant.
def is_vacant(board, row, col, piece):
    # print("Check vacant")
    return board[row][col] == 0

# Determine if the placement of piece will lead to any reversals
def is_reversible(board, row, col, piece):
    # print("Determine Reversible")

    # Check right:
    if (col+1) <= DIM:
        # print(" determine right")
        for c in range(col+1, DIM): # Start from the one to the right, not itself
            if c == (col+1):
                if board[row][c] == 0 or board[row][c] == piece:
                    break
            else:
                if board[row][c] == 0:
                    break
                if board[row][c] == piece:
                    return True
    # Check left (must check from right to left):
    if (col-1) >= 0:
        # print(" determine left")
        for c in range(col-1, 0, -1):
            if c == (col-1):
                if board[row][c] == 0 or board[row][c] == piece:
                    break
            else:
                if board[row][c] == 0:
                    break
                if board[row][c] == piece:
                    return True

    # Check up (must check from down to up):
    if (row-1) >= 0:
        # print(" determine up")
        for r in range(row-1, 0, -1):
            if r==(row-1):
                if board[r][col] == 0 or board[r][col] == piece:
                    break
            else:
                if board[r][col] == 0:
                    break
                if board[r][col] == piece:
                    return True

    # Check down:
    if (row+1) <= DIM:
        # print(" determine down")
        for r in range(row+1, DIM):
            if r == (row+1):
                if board[r][col] == 0 or board[r][col] == piece:
                    break
            else:
                if board[r][col] == 0:
                    break
                if board[r][col] == piece:
                    return True

    # Check positive diagonal, left of chess (going up to the right, so rows decreasing):
    if (col-1) >= 0:
        row_it = row+1
        # print(" determine +ve diagonal left")
        for c in range(col-1, 0, -1):
            if (row_it) >= DIM:
                break
            if c==(col-1):
                if board[row_it][c] == 0 or board[row_it][c] == piece:
                    break
            else:
                if board[row_it][c] == 0:
                    break
                if board[row_it][c] == piece:
                    # print("     location: row=", row_it, ", col=",c)
                    return True
            row_it = row_it + 1

    # Check positive diagonal, right of chess:
    if (col+1) <= DIM:
        row_it = row-1
        # print(" determine +ve diagonal right")
        for c in range(col+1, DIM):
            if (row_it) < 0:
                break
            if c==(col+1):
                if board[row_it][c] == 0 or board[row_it][c] == piece:
                    break
            else:
                if board[row_it][c] == 0:
                    break
                if board[row_it][c] == piece:
                    return True
            row_it = row_it - 1

    # Check negative diagonal, left of chess:
    if (col-1) >= 0:
        row_it = row-1
        # print(" determine -ve diagonal left")
        for c in range(col-1, 0, -1):
            if (row_it) < 0:
                break
            if c==(col-1):
                if board[row_it][c] == 0 or board[row_it][c] == piece:
                    break
            else:
                if board[row_it][c] == 0:
                    break
                if board[row_it][c] == piece:
                    return True
            row_it = row_it - 1

    # Check negative diagonal, right of chess:
```

```
    if (col+1) <= DIM:
        row_it = row+1
        # print(" determine -ve diagonal right")
        for c in range(col+1, DIM):
            if (row_it) >= DIM:
                break
            if c==(col+1):
                if board[row_it][c] == 0 or board[row_it][c] == piece:
                    break
            else:
                if board[row_it][c] == 0:
                    break
                if board[row_it][c] == piece:
                    return True
            row_it = row_it + 1

    # Return false if cannot find any reversible pieces
    return False

# Drop piece, find nearest piece (with opponent in between) in vert/horz/diag axis and reverse the pieces
def drop_piece(board, row, col, piece):

    flip_num = 0

    # print("Drop piece and reverse")
    board[row][col] = piece

    # Variables
    reverse = False
    opp_row = row # identified row of the other chess
    opp_col = col # identified column of the other chess

    # Reverse pieces on the right:
    if (col+1) <= DIM:
        # print ("    check right", row, col)
        for c in range(col+1, DIM): # Start from the one to the right, not itself
            if board[row][c] == 0:
                break
            if board[row][c] == piece:
                reverse = True
                opp_col = c
                break
        if reverse == True:
            for c in range(col+1, opp_col):
                board[row][c] = piece
                flip_num = flip_num+1
                # print("         reverse piece at row=", row, ", col=",c)
        reverse = False

    # Reverse left (must check from right to left):
    if (col-1) >= 0:
        # print ("    check left", row, col)
        for c in range(col-1, 0, -1):
            if board[row][c] == 0:
                break
            if board[row][c] == piece:
                reverse = True
                opp_col = c
                break
        if reverse == True:
            for c in range(col-1, opp_col, -1):
                board[row][c] = piece
                flip_num = flip_num+1
                # print("         reverse piece at row=", row, ", col=",c)
        reverse = False

    # Reverse up (must check from down to up):
    if (row-1) >= 0:
        # print ("    check up", row, col)
        for r in range(row-1, 0, -1):
            if board[r][col] == 0:
                break
            if board[r][col] == piece:
                reverse = True
                opp_row = r
                break
        if reverse == True:
            for r in range(row-1, opp_row, -1):
                board[r][col] = piece
                flip_num = flip_num+1
                # print("         reverse piece at row=", r, ", col=",col)
        reverse = False

    # Reverse down:
    if (row-1) <= DIM:
        # print ("    check down", row, col)
        for r in range(row+1, DIM):
            if board[r][col] == 0:
                break
            if board[r][col] == piece:
                reverse = True
                opp_row = r
```

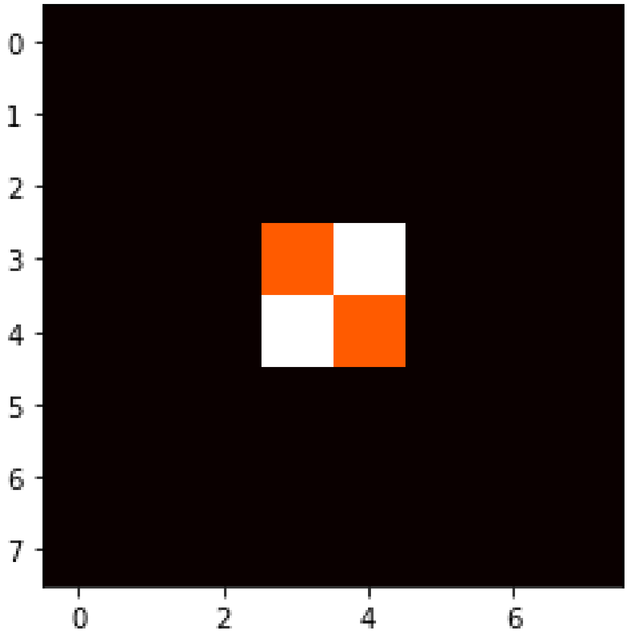
```python
                break
        if reverse == True:
            for r in range(row+1, opp_row):
                board[r][col] = piece
                flip_num = flip_num+1
                # print("         reverse piece at row=", r, ", col=",col)
        reverse = False

    # Reverse positive diagonal, left of chess (going up to the right, so rows decreasing):
    if (col-1) >= 0:
        # print ("    check positive diagonal left", row, col)
        row_it = row+1
        for c in range(col-1, 0, -1):
            if (row_it) >= DIM or board[row_it][c] == 0:
                break
            if board[row_it][c] == piece:
                reverse = True
                opp_row = row_it
                opp_col = c
                break
            row_it = row_it + 1
        if reverse == True:
            row_it = row+1
            for c in range (col-1, opp_col, -1):
                board[row_it][c] = piece
                flip_num = flip_num+1
                # print("         reverse piece at row=", row_it, ", col=",c)
                row_it = row_it + 1
        reverse = False

    # Reverse positive diagonal, right of chess:
    if (col+1) <= DIM:
        # print ("    check positive diagonal right", row, col)
        row_it = row-1
        for c in range(col+1, DIM):
            if (row_it) < 0 or board[row_it][c] == 0:
                break
            if board[row_it][c] == piece:
                reverse = True
                opp_row = row_it
                opp_col = c
                break
            row_it = row_it - 1
        if reverse == True:
            row_it = row-1
            for c in range (col+1, opp_col):
                board[row_it][c] = piece
                flip_num = flip_num+1
                # print("         reverse piece at row=", row_it, ", col=",c)
                row_it = row_it - 1
        reverse = False

    # Reverse negative diagonal, left of chess:
    if (col-1) >= 0:
        # print ("    check negative diagonal left", row, col)
        row_it = row-1
        for c in range(col-1, 0, -1):
            if (row_it) < 0 or board[row_it][c] == 0:
                break
            if board[row_it][c] == piece:
                reverse = True
                opp_row = row_it
                opp_col = c
                break
            row_it = row_it - 1
        if reverse == True:
            row_it = row-1
            for c in range (col-1, opp_col, -1):
                board[row_it][c] = piece
                flip_num = flip_num+1
                # print("         reverse piece at row=", row_it, ", col=",c)
                row_it = row_it - 1
        reverse = False

    # Reverse negative diagonal, right of chess:
    if (col+1) <= DIM:
        # print ("    check negative diagonal right", row, col)
        row_it = row+1
        for c in range(col+1, DIM):
            if (row_it) >= DIM or board[row_it][c] == 0:
                break
            if board[row_it][c] == piece:
                reverse = True
                opp_row = row_it
                opp_col = c
                break
            row_it = row_it + 1
        if reverse == True:
            row_it = row+1
            for c in range (col+1, opp_col):
                board[row_it][c] = piece
                flip_num = flip_num+1
                # print("         reverse piece at row=", row_it, ", col=",c)
                row_it = row_it + 1
        reverse = False

    return flip_num

# Print board and scores
def print_board(board, flip_num): #
    p1_score = np.count_nonzero(board==1)
    p2_score = np.count_nonzero(board==2)
    print("\nPlayer 1 pieces =", p1_score)
    print("Player 2 pieces =", p2_score)
    print("Number of flipped pieces: ", flip_num)
    print("Total number of pieces on the board: ", np.count_nonzero(board))
    plt.imshow(board, cmap='hot', interpolation='nearest')
    plt.show()

# End game (True if all entries are filled in)
def is_end_game(board):
    for c in range(DIM):
        for r in range(DIM):
            if board[r][c] == 0:
                return False
    return True

# For note book only
def insert(row, col, piece):
    flip_num = drop_piece(board, row, col, piece)
    print_board(board, flip_num)
```

Creating a board……

[2]:

```
board = create_board()
print_board(board, 0)
```

```
Player 1 pieces = 2
Player 2 pieces = 2
Number of flipped pieces:  0
Total number of pieces on the board:  4
```

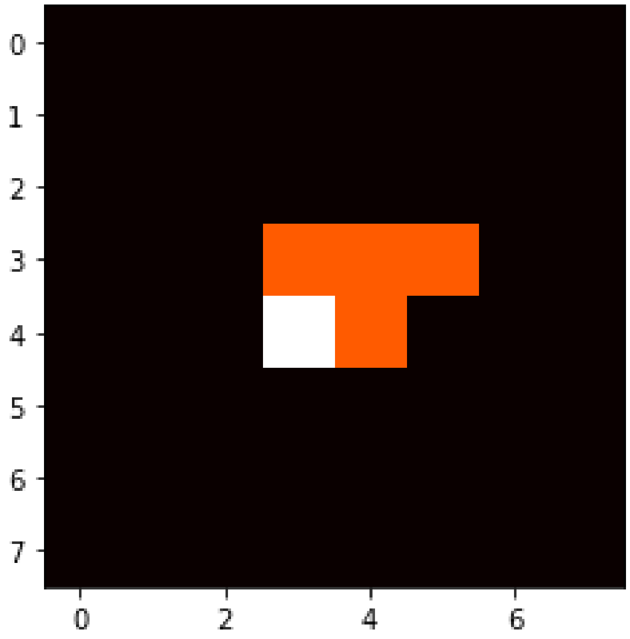


1.2 Let the game start!

This is a full documentation of a game that I've played with myself (yeah……) Hopefully there's nothing wrong by now……

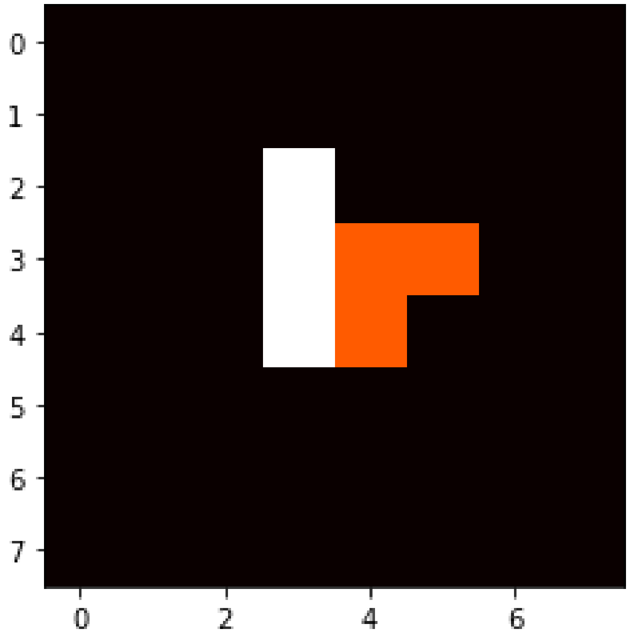
[3]:
```
insert(3, 5, 1)
```

```
Player 1 pieces = 4
Player 2 pieces = 1
Number of flipped pieces:  1
Total number of pieces on the board:  5
```



```
[4]: insert(2, 3, 2)
```

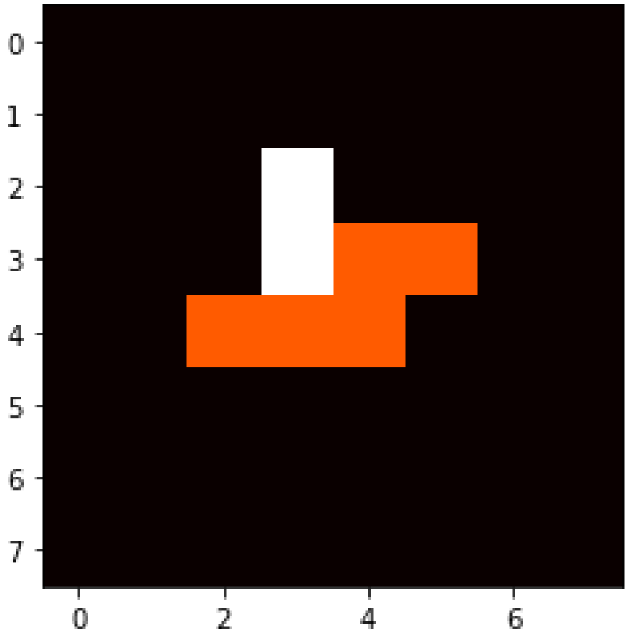
```
Player 1 pieces = 3
Player 2 pieces = 3
Number of flipped pieces:  1
Total number of pieces on the board:  6
```



[5]:

```
insert(4, 2, 1)
```

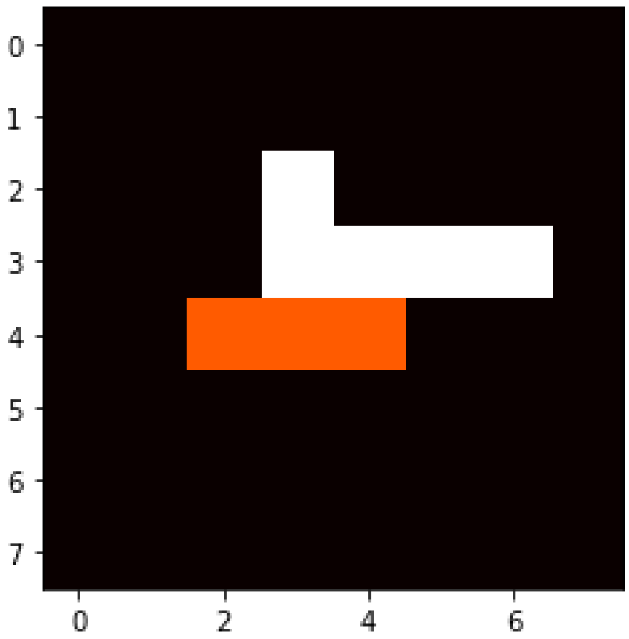
```
Player 1 pieces = 5
Player 2 pieces = 2
Number of flipped pieces:  1
Total number of pieces on the board:  7
```



[6]:

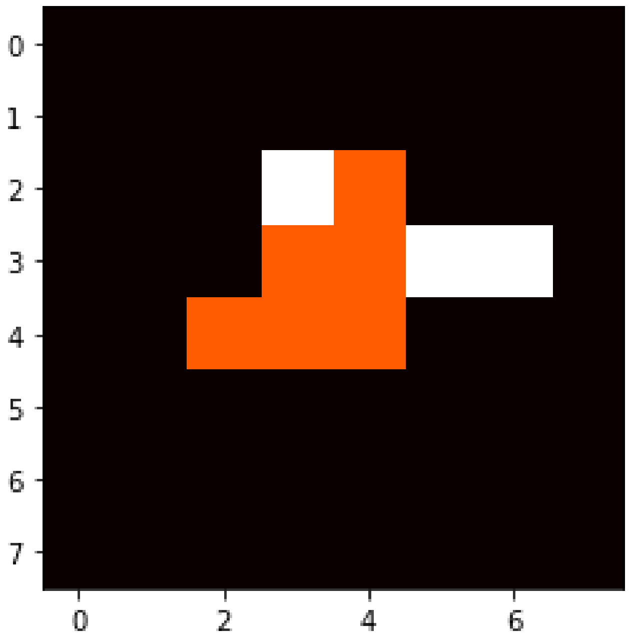
```
insert(3, 6, 2)
```

```
Player 1 pieces = 3
Player 2 pieces = 5
Number of flipped pieces:  2
Total number of pieces on the board:  8
```



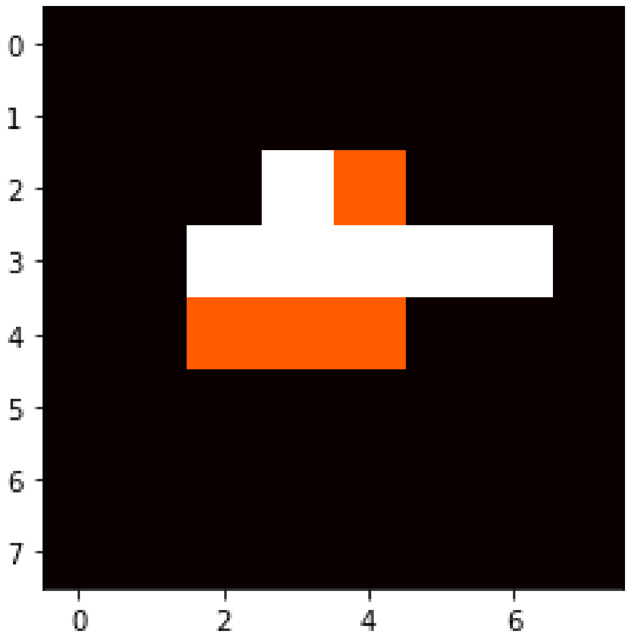
[7]:
```
insert(2, 4, 1)
```

```
Player 1 pieces = 6
Player 2 pieces = 3
Number of flipped pieces:  2
Total number of pieces on the board:  9
```



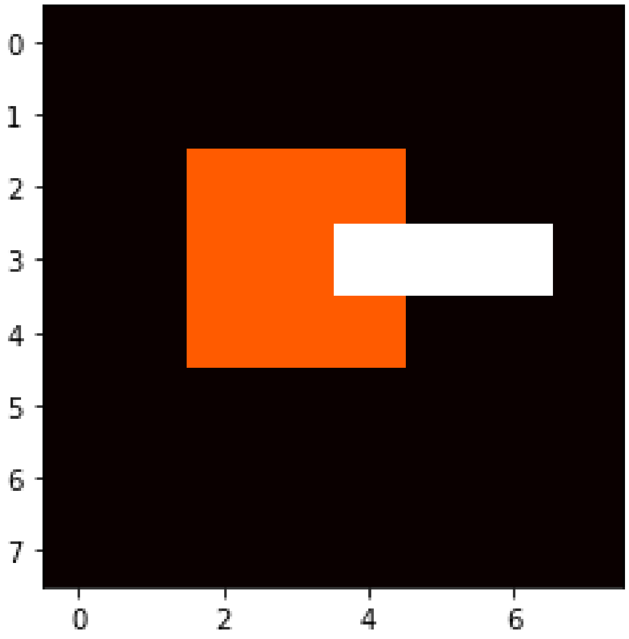
[8]:
```
insert(3, 2, 2)
```

```
Player 1 pieces = 4
Player 2 pieces = 6
Number of flipped pieces:  2
Total number of pieces on the board:  10
```



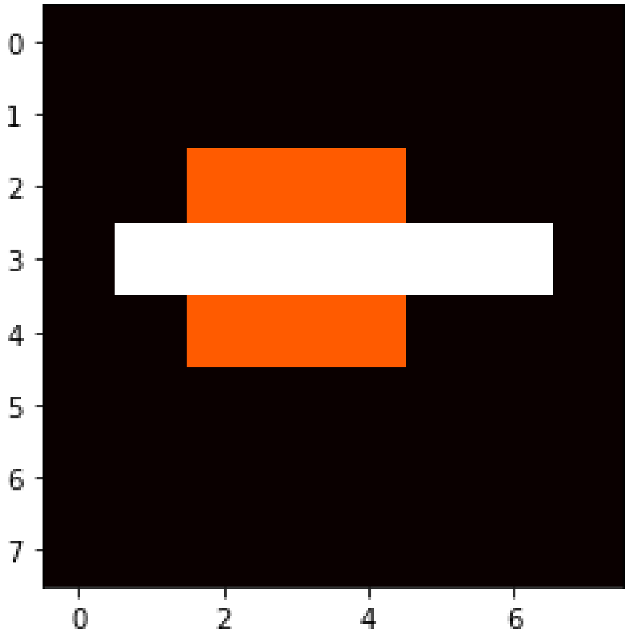
[9]:
```
insert(2,2,1)
```

```
Player 1 pieces = 8
Player 2 pieces = 3
Number of flipped pieces:  3
Total number of pieces on the board:  11
```



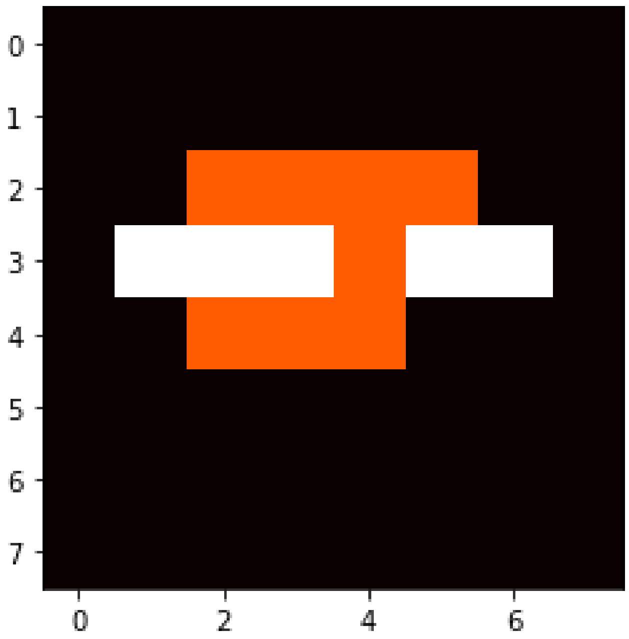
[10]:
```
insert(3,1,2)
```

```
Player 1 pieces = 6
Player 2 pieces = 6
Number of flipped pieces:  2
Total number of pieces on the board:  12
```



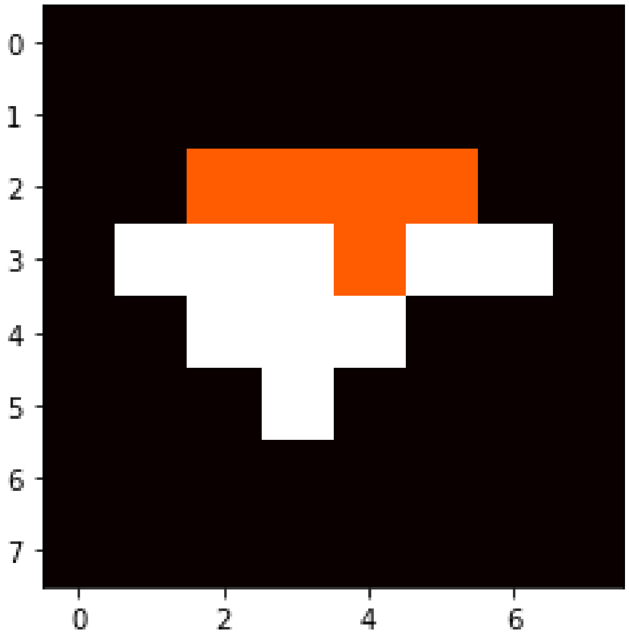
[11]:
```
insert(2,5,1)
```

```
Player 1 pieces = 8
Player 2 pieces = 5
Number of flipped pieces:  1
Total number of pieces on the board:  13
```



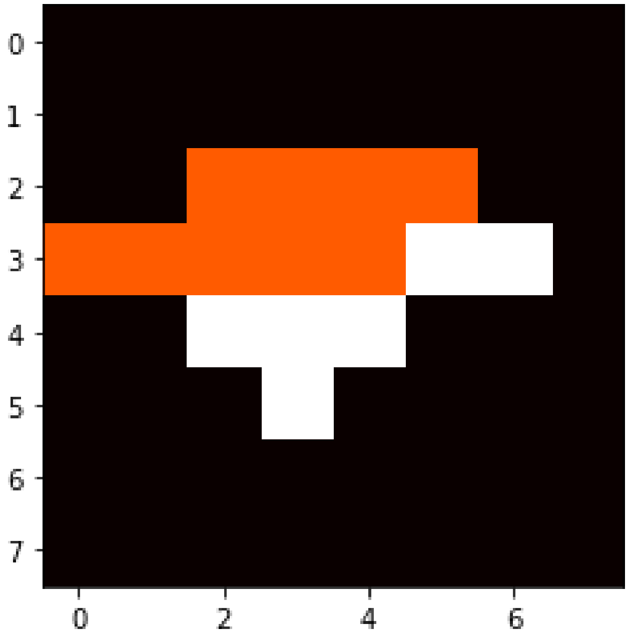
[12]:
```
insert(5,3,2)
```

```
Player 1 pieces = 5
Player 2 pieces = 9
Number of flipped pieces:  3
Total number of pieces on the board:  14
```



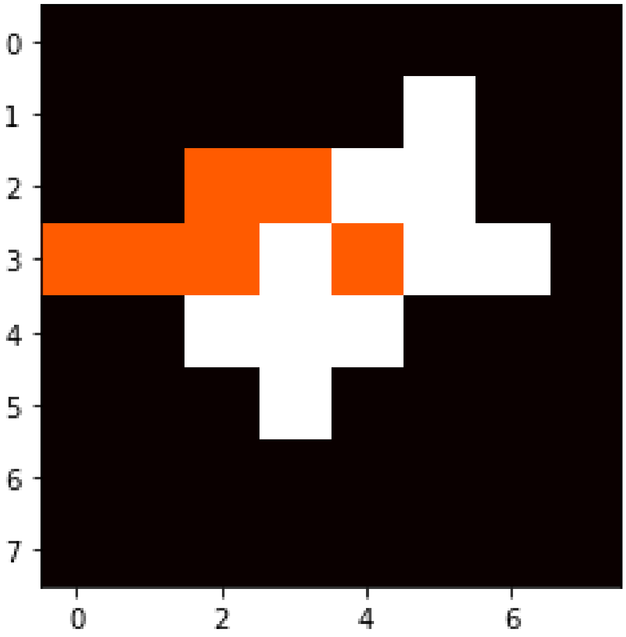
```
[13]: insert(3,0,1)
```

```
Player 1 pieces = 9
Player 2 pieces = 6
Number of flipped pieces:  3
Total number of pieces on the board:  15
```



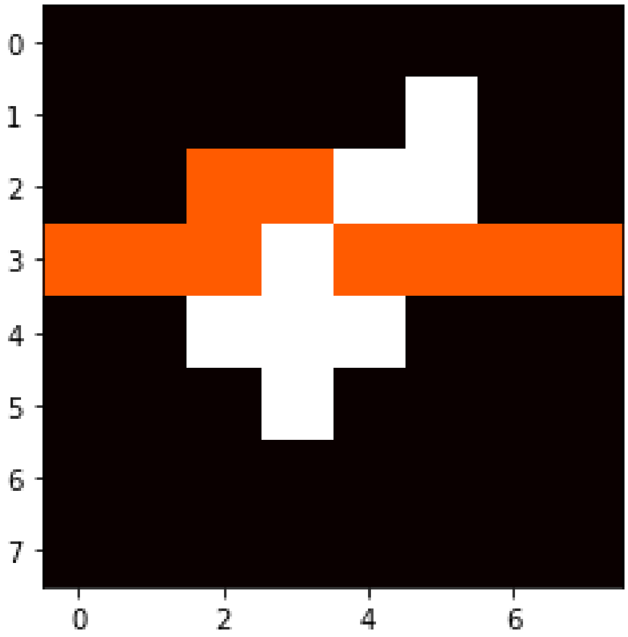
[14]:
```
insert(1,5,2)
```

```
Player 1 pieces = 6
Player 2 pieces = 10
Number of flipped pieces:  3
Total number of pieces on the board:  16
```



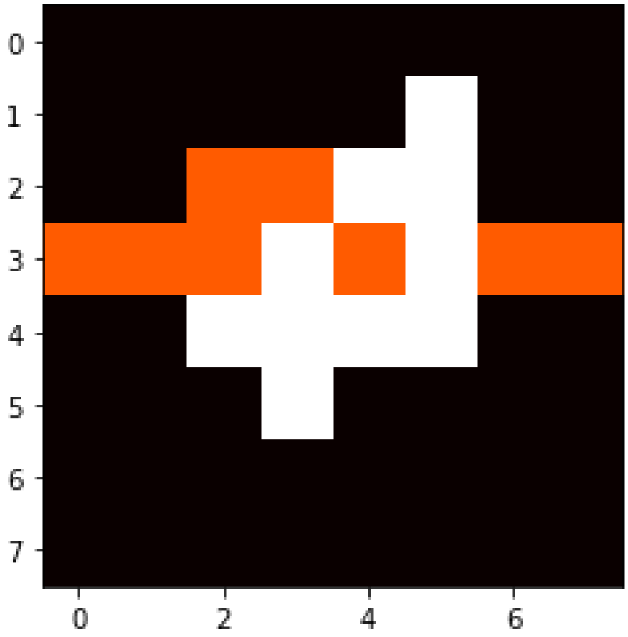
[15]:
```
insert(3,7,1)
```

```
Player 1 pieces = 9
Player 2 pieces = 8
Number of flipped pieces:  2
Total number of pieces on the board:  17
```



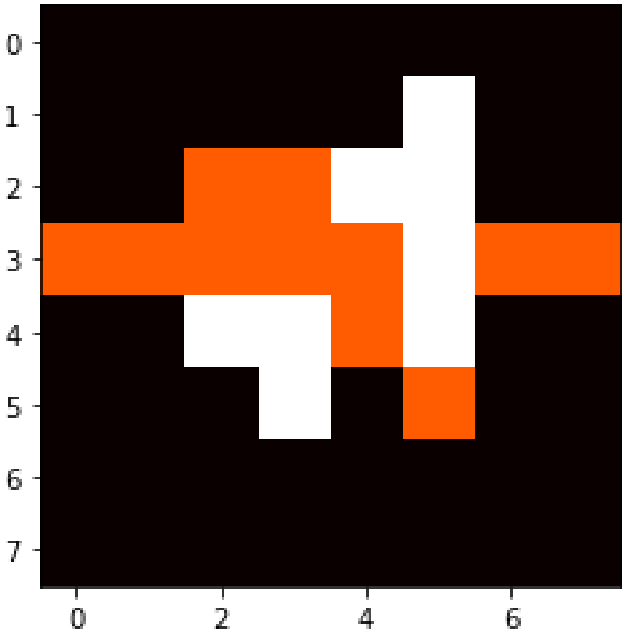
```
[16]: insert(4,5,2)
```

```
Player 1 pieces = 8
Player 2 pieces = 10
Number of flipped pieces:  1
Total number of pieces on the board:  18
```



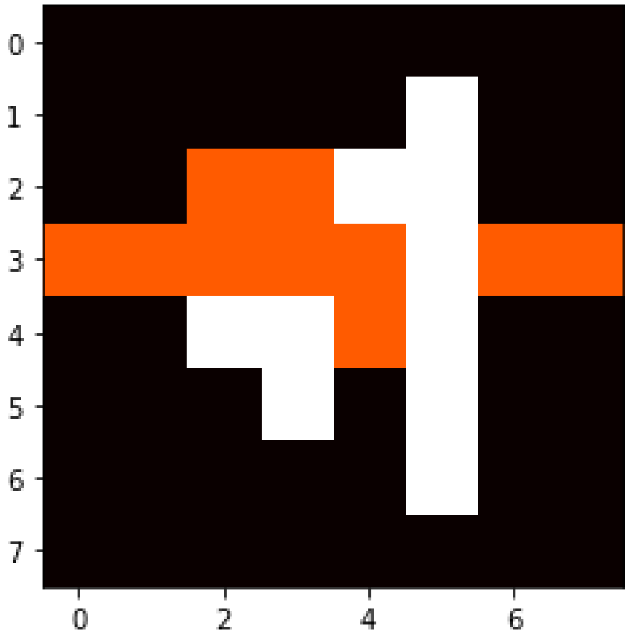
```
[17]: insert(5,5,1)
```

```
Player 1 pieces = 11
Player 2 pieces = 8
Number of flipped pieces:  2
Total number of pieces on the board:  19
```



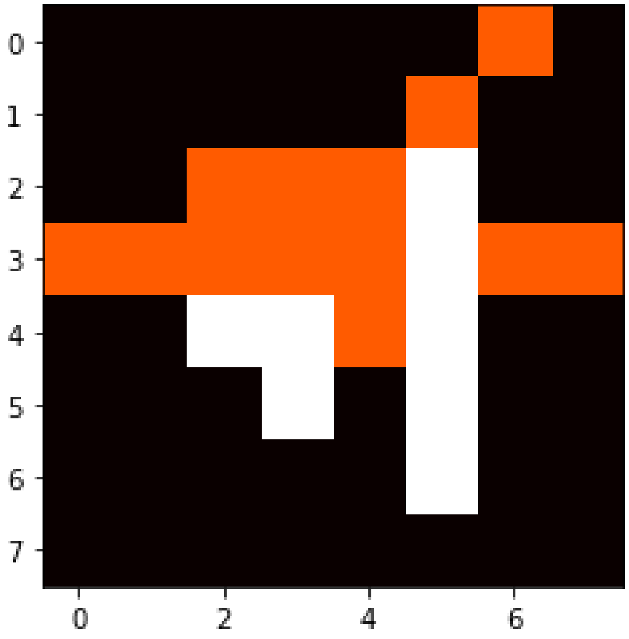
```
[18]: insert(6,5,2)
```

```
Player 1 pieces = 10
Player 2 pieces = 10
Number of flipped pieces:  1
Total number of pieces on the board:  20
```



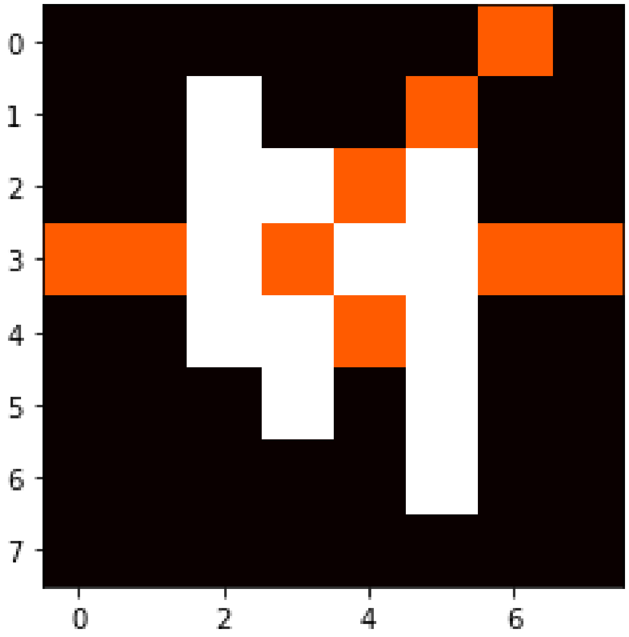
[19]:
```
insert(0,6,1)
```

```
Player 1 pieces = 13
Player 2 pieces = 8
Number of flipped pieces:  2
Total number of pieces on the board:  21
```



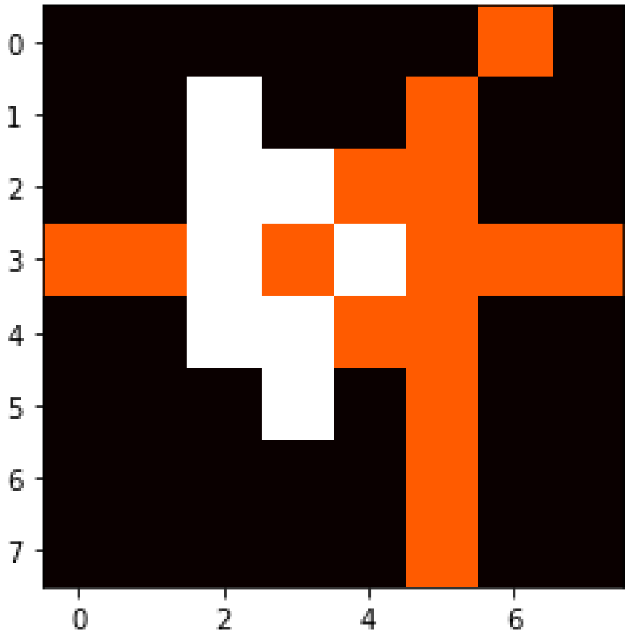
[20]:
```
insert(1,2,2)
```

```
Player 1 pieces = 9
Player 2 pieces = 13
Number of flipped pieces:  4
Total number of pieces on the board:  22
```



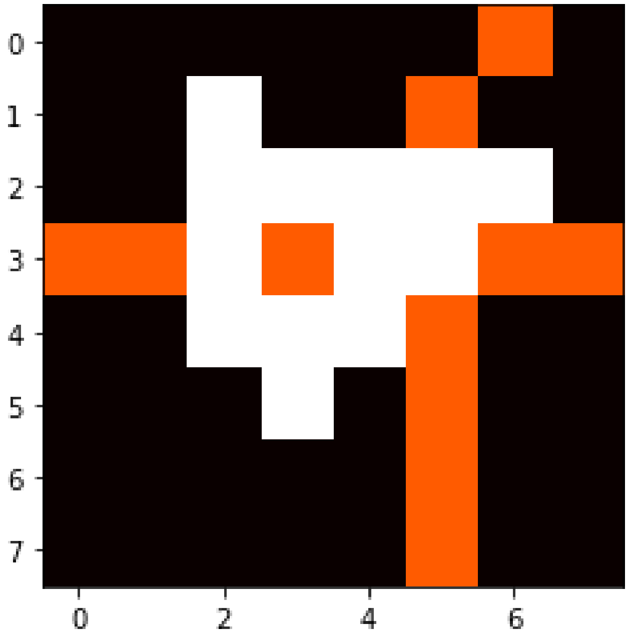
[21]:
```
insert(7,5,1)
```

```
Player 1 pieces = 15
Player 2 pieces = 8
Number of flipped pieces:  5
Total number of pieces on the board:  23
```



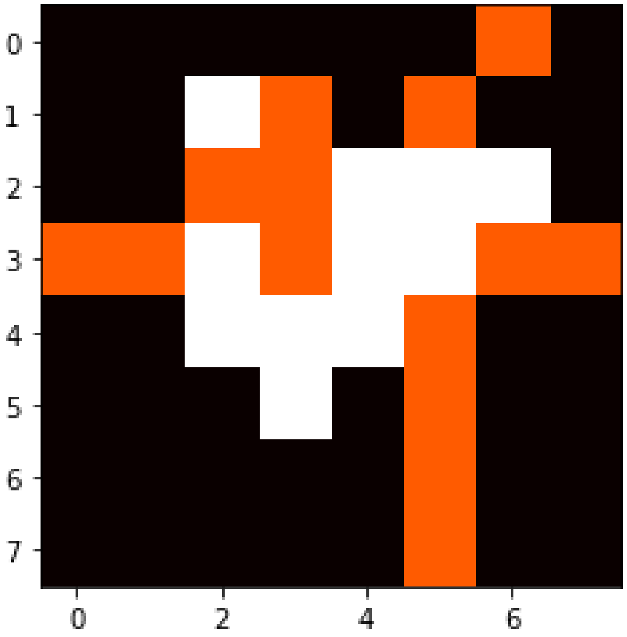
[22]:
```
insert(2,6,2)
```

```
Player 1 pieces = 11
Player 2 pieces = 13
Number of flipped pieces:  4
Total number of pieces on the board:  24
```



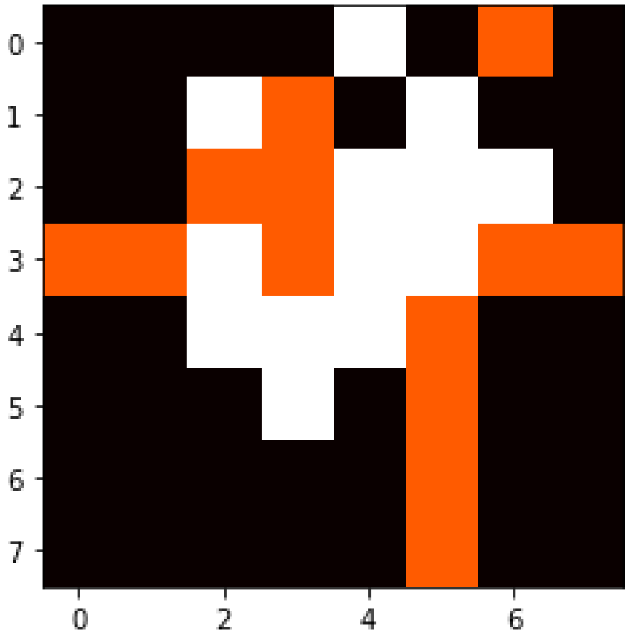
[23]:
```
insert(1,3,1)
```

```
Player 1 pieces = 14
Player 2 pieces = 11
Number of flipped pieces:  2
Total number of pieces on the board:  25
```



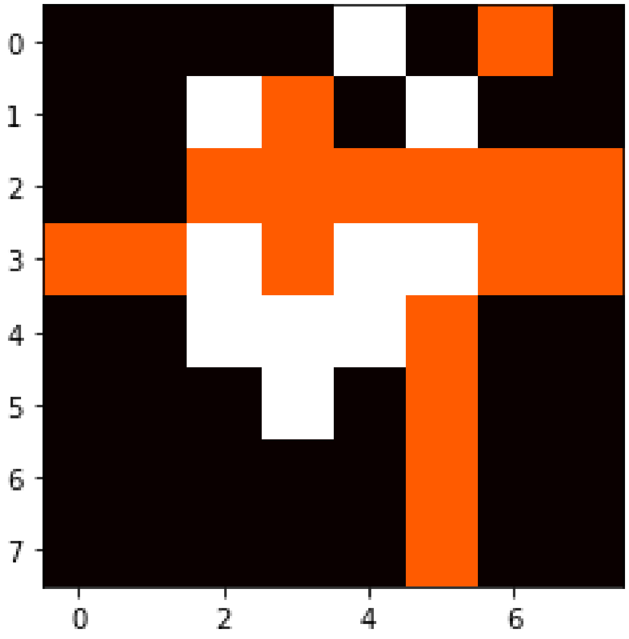
[24]:
```
insert(0,4,2)
```

```
Player 1 pieces = 13
Player 2 pieces = 13
Number of flipped pieces:  1
Total number of pieces on the board:  26
```



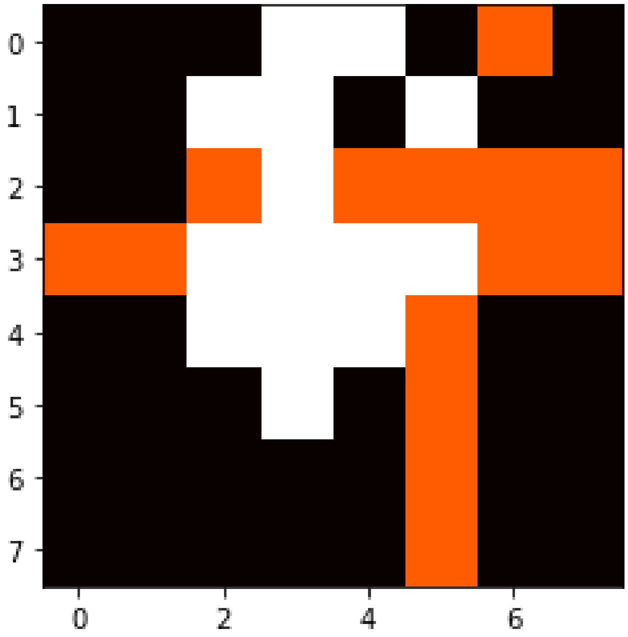
```
[25]: insert(2,7,1)
```

```
Player 1 pieces = 17
Player 2 pieces = 10
Number of flipped pieces:  3
Total number of pieces on the board:  27
```



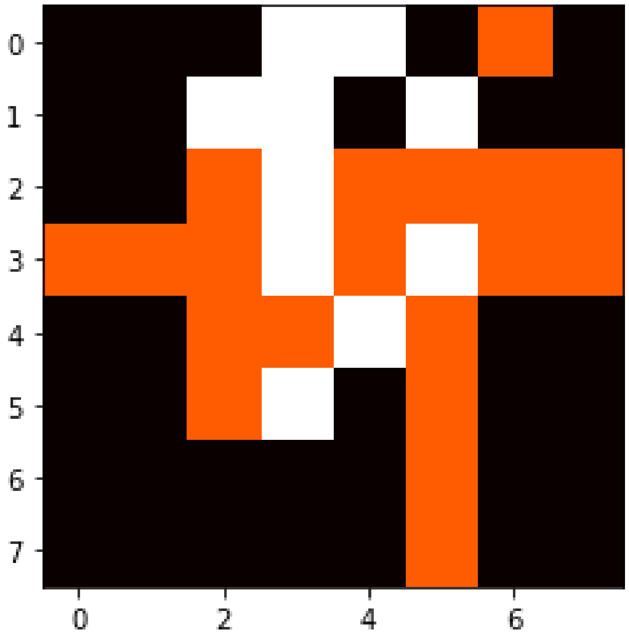
[26]:
```
insert(0,3,2)
```

```
Player 1 pieces = 14
Player 2 pieces = 14
Number of flipped pieces:  3
Total number of pieces on the board:  28
```



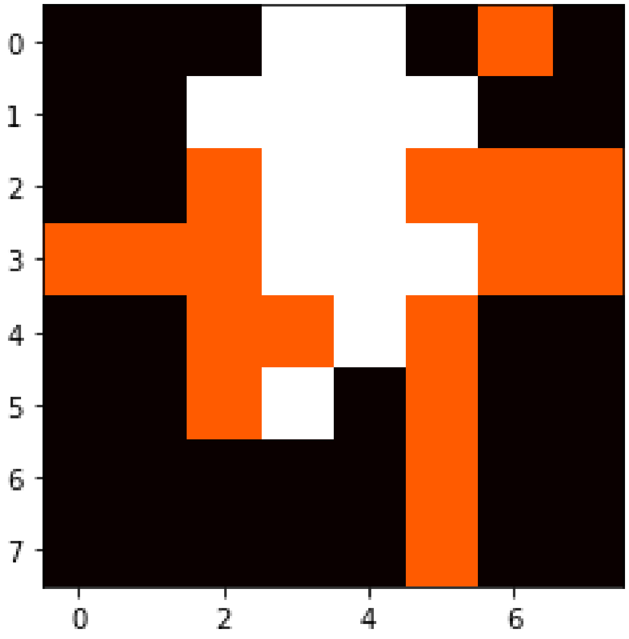
[27]:
```
insert(5,2,1)
```

```
Player 1 pieces = 19
Player 2 pieces = 10
Number of flipped pieces:  4
Total number of pieces on the board:  29
```



[28]:
```
insert(1,4,2)
```

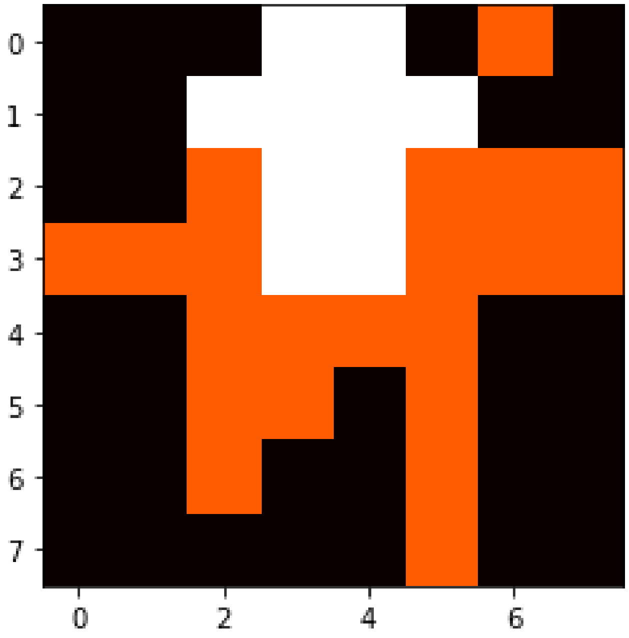
```
Player 1 pieces = 17
Player 2 pieces = 13
Number of flipped pieces:  2
Total number of pieces on the board:  30
```



[29]:

```
insert(6,2,1)
```

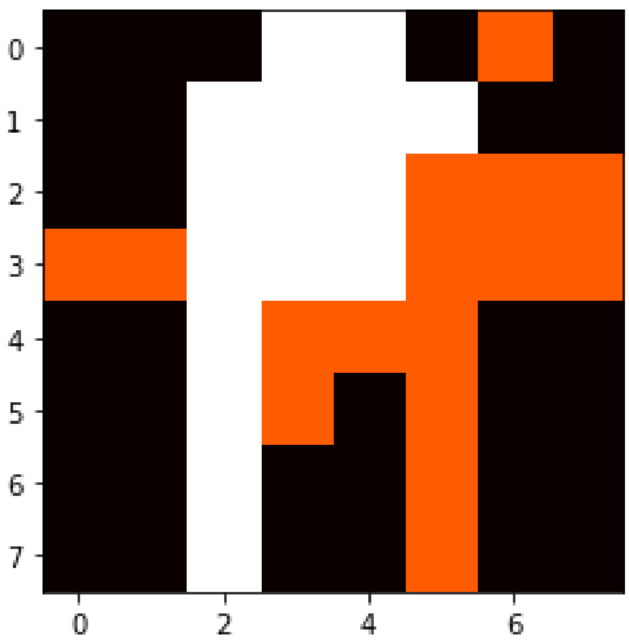
```
Player 1 pieces = 21
Player 2 pieces = 10
Number of flipped pieces:  3
Total number of pieces on the board:  31
```



[30]:

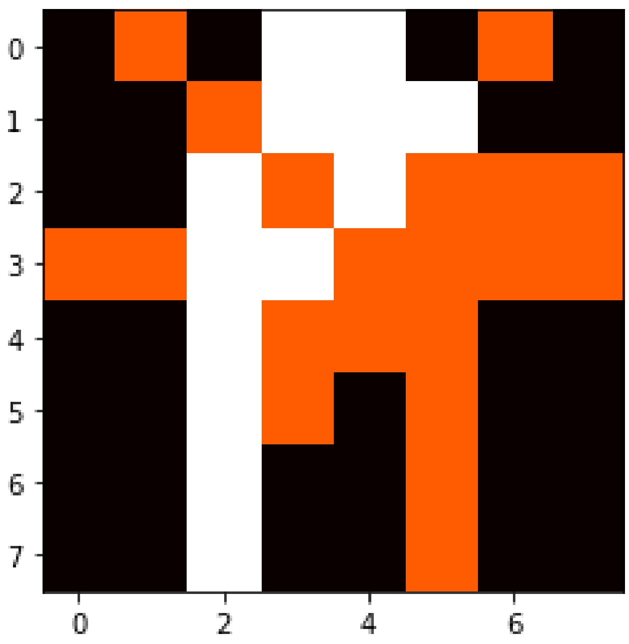
```
insert(7,2,2)
```

```
Player 1 pieces = 16
Player 2 pieces = 16
Number of flipped pieces:  5
Total number of pieces on the board:  32
```



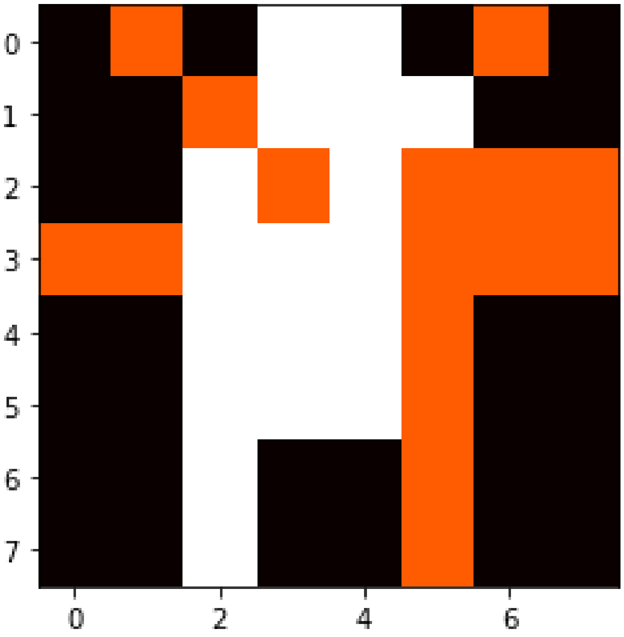
```
[31]: insert(0,1,1)
```

```
Player 1 pieces = 20
Player 2 pieces = 13
Number of flipped pieces:  3
Total number of pieces on the board:  33
```



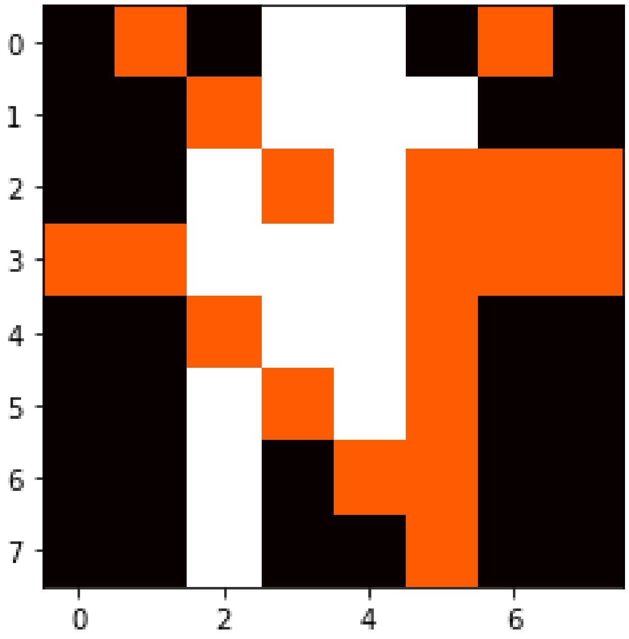
```
[32]: insert(5,4,2)
```

```
Player 1 pieces = 16
Player 2 pieces = 18
Number of flipped pieces:  4
Total number of pieces on the board:  34
```



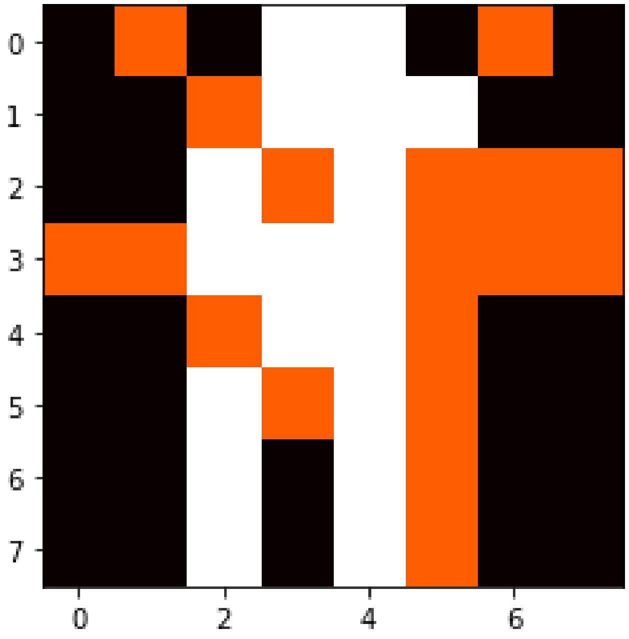
```
[33]: insert(6,4,1)
```

```
Player 1 pieces = 19
Player 2 pieces = 16
Number of flipped pieces:  2
Total number of pieces on the board:  35
```



[34]:
```
insert(7,4,2)
```

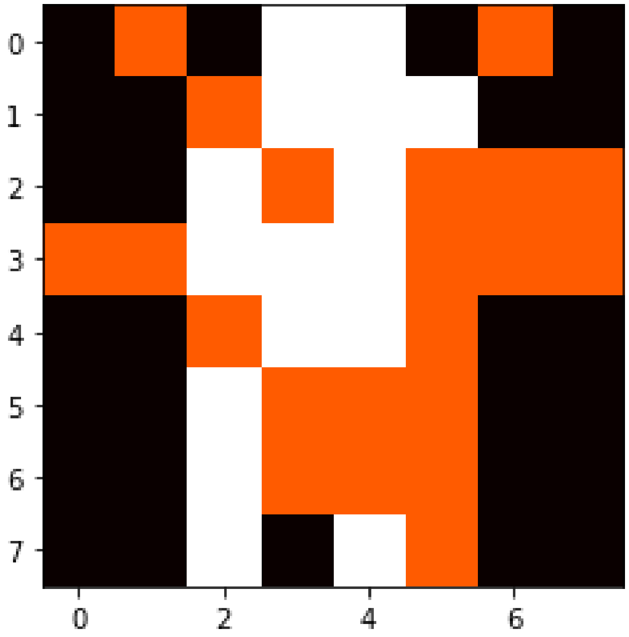
```
Player 1 pieces = 18
Player 2 pieces = 18
Number of flipped pieces:  1
Total number of pieces on the board:  36
```



[35]:

```
insert(6,3,1)
```

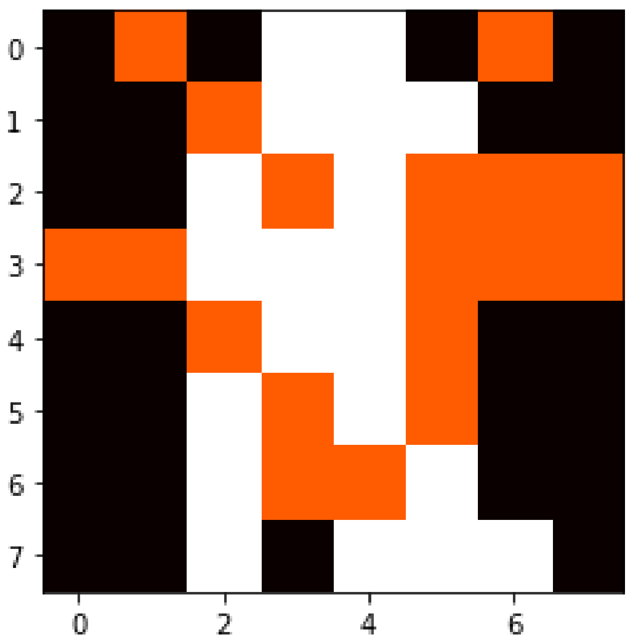
```
Player 1 pieces = 21
Player 2 pieces = 16
Number of flipped pieces:  2
Total number of pieces on the board:  37
```



[36]:

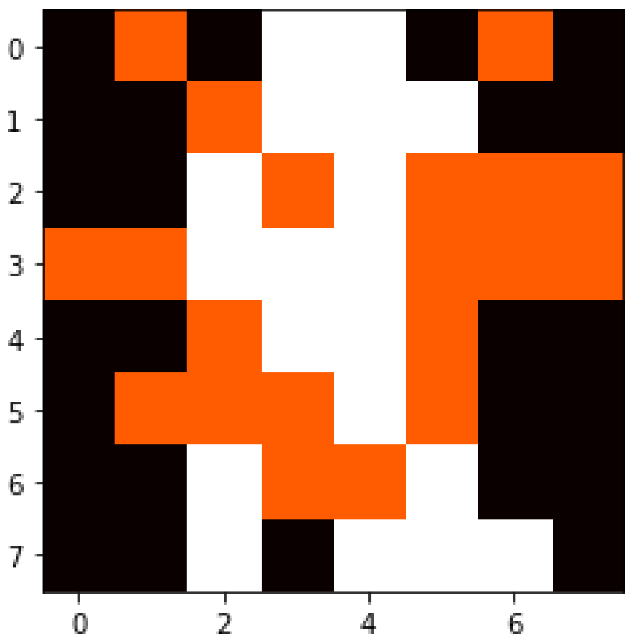
```
insert(7,6,2)
```

```
Player 1 pieces = 18
Player 2 pieces = 20
Number of flipped pieces:  3
Total number of pieces on the board:  38
```



[37]:
```
insert(5,1,1)
```

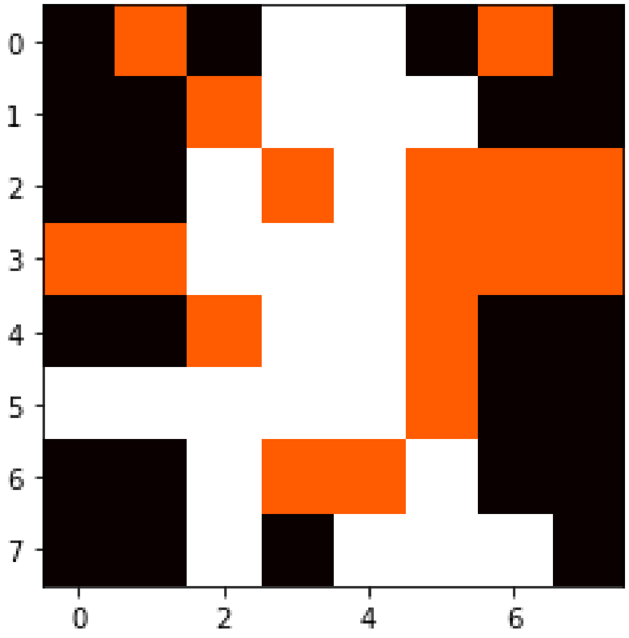
```
Player 1 pieces = 20
Player 2 pieces = 19
Number of flipped pieces:  1
Total number of pieces on the board:  39
```



[38]:

```
insert(5,0,2)
```

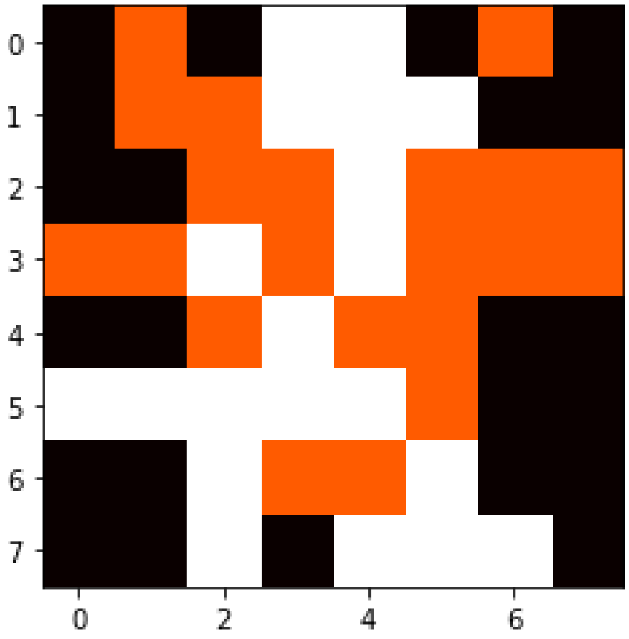
```
Player 1 pieces = 17
Player 2 pieces = 23
Number of flipped pieces:  3
Total number of pieces on the board:  40
```



[39]:

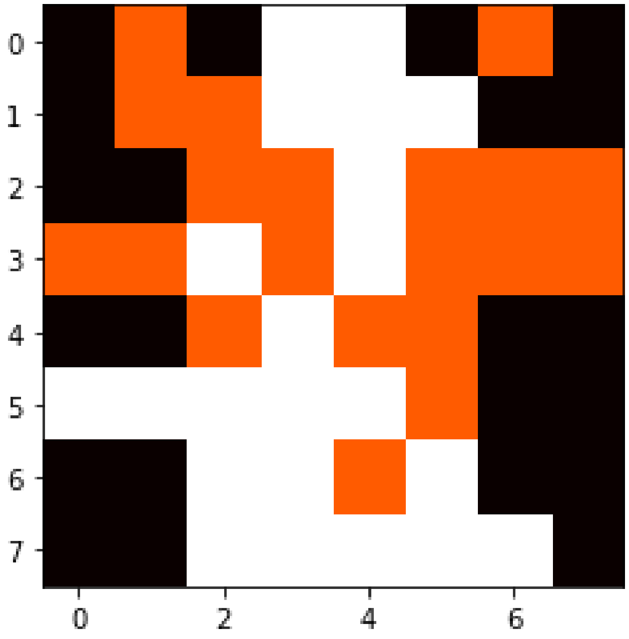
```
insert(1,1,1)
```

```
Player 1 pieces = 21
Player 2 pieces = 20
Number of flipped pieces:  3
Total number of pieces on the board:  41
```



[40]:
```
insert(7,3,2)
```

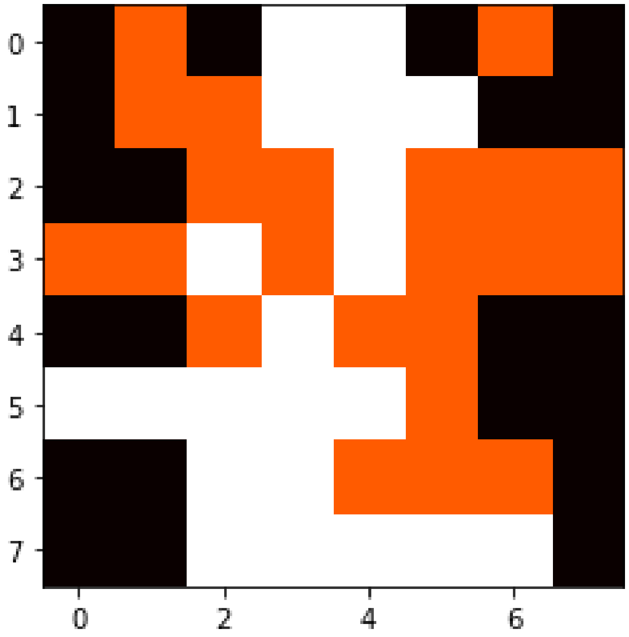
```
Player 1 pieces = 20
Player 2 pieces = 22
Number of flipped pieces:  1
Total number of pieces on the board:  42
```



[41]:

```
insert(6,6,1)
```

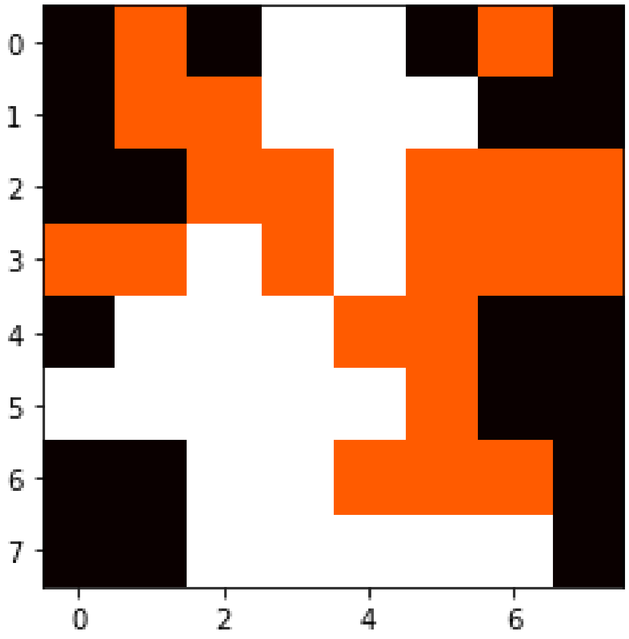
```
Player 1 pieces = 22
Player 2 pieces = 21
Number of flipped pieces:  1
Total number of pieces on the board:  43
```



[42]:

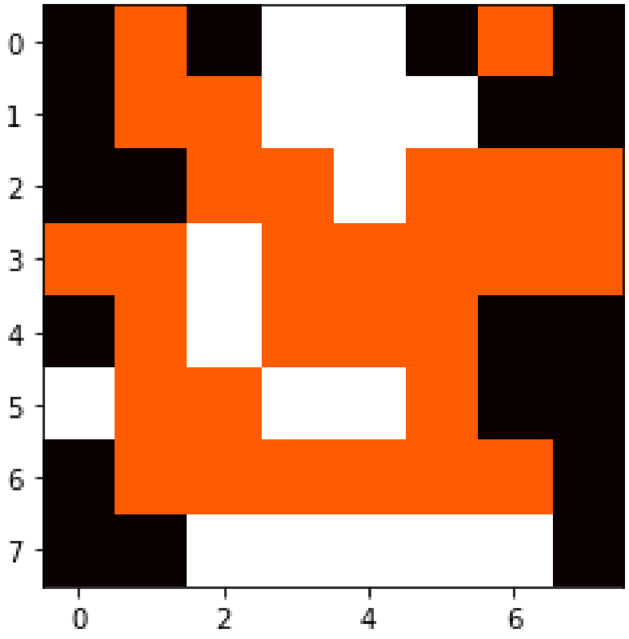
```
insert(4,1,2)
```

```
Player 1 pieces = 21
Player 2 pieces = 23
Number of flipped pieces:  1
Total number of pieces on the board:  44
```



```
[43]: insert(6,1,1)
```

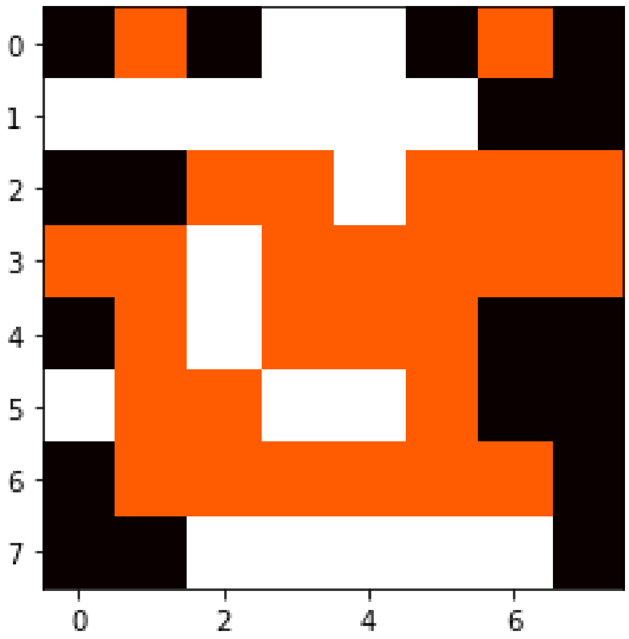
```
Player 1 pieces = 29
Player 2 pieces = 16
Number of flipped pieces:  7
Total number of pieces on the board:  45
```



[44]:

```
insert(1,0,2)
```

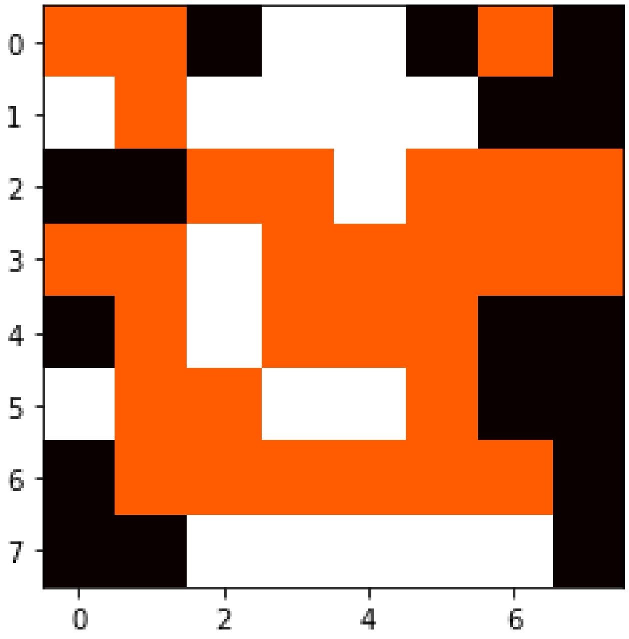
```
Player 1 pieces = 27
Player 2 pieces = 19
Number of flipped pieces:  2
Total number of pieces on the board:  46
```



[45]:

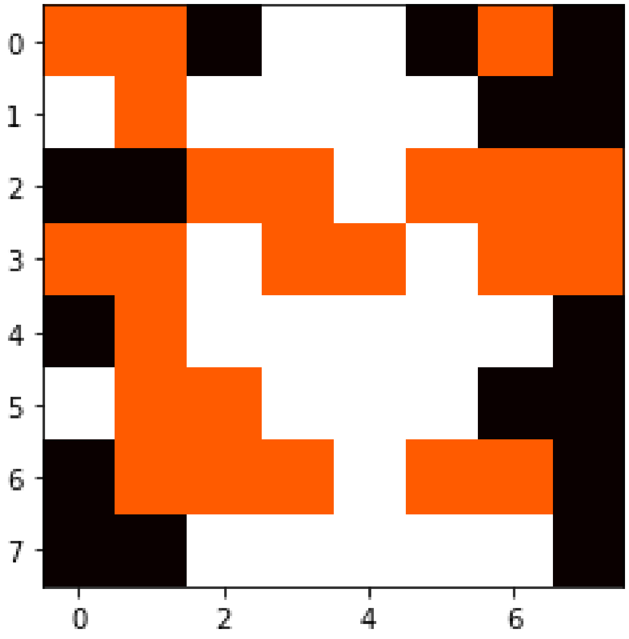
```
insert(0,0,1)
```

```
Player 1 pieces = 29
Player 2 pieces = 18
Number of flipped pieces:  1
Total number of pieces on the board:  47
```



[46]:
```
insert(4,6,2)
```

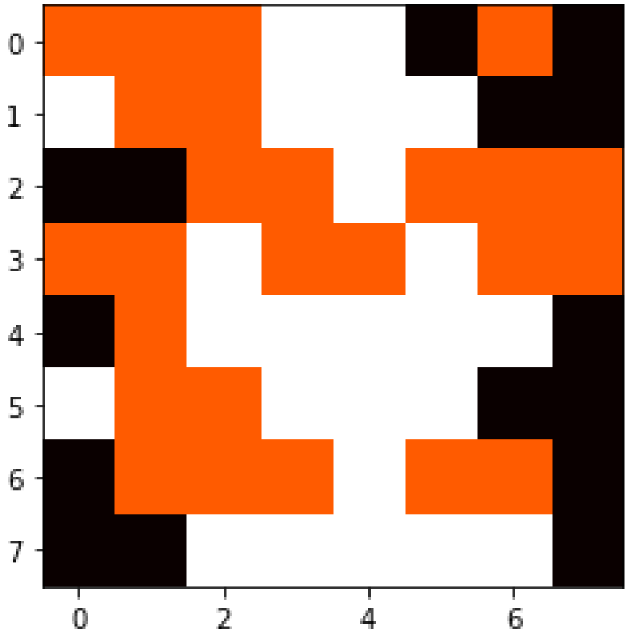
```
Player 1 pieces = 23
Player 2 pieces = 25
Number of flipped pieces:  6
Total number of pieces on the board:  48
```



[47]:

```
insert(0,2,1)
```

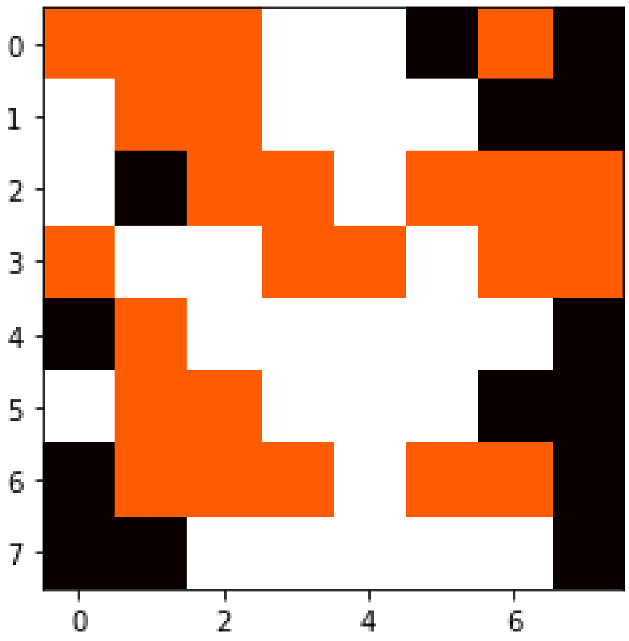
```
Player 1 pieces = 25
Player 2 pieces = 24
Number of flipped pieces:  1
Total number of pieces on the board:  49
```



[48]:

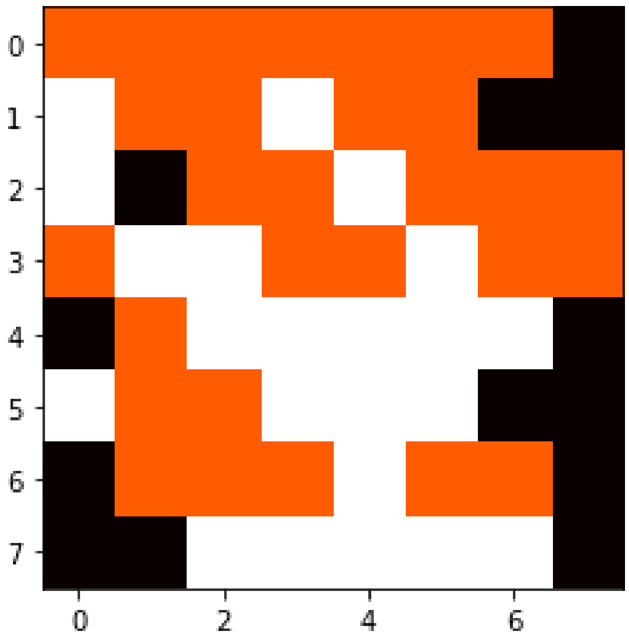
```
insert(2,0,2)
```

```
Player 1 pieces = 24
Player 2 pieces = 26
Number of flipped pieces:  1
Total number of pieces on the board:  50
```



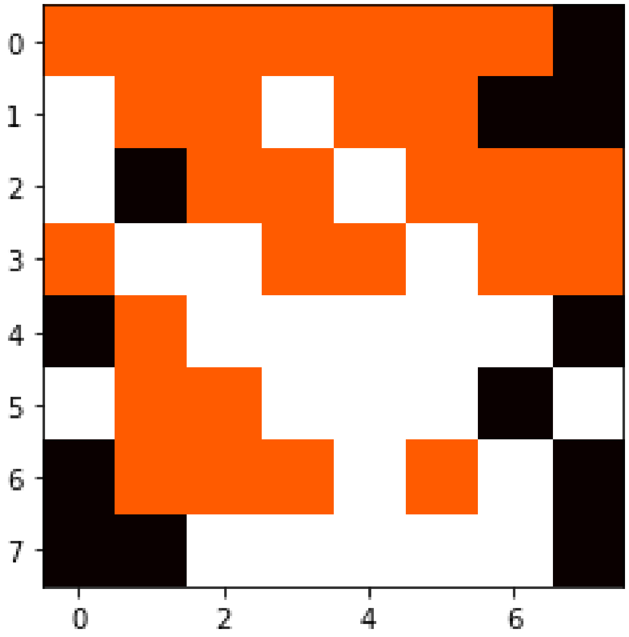
[49]:
```
insert(0,5,1)
```

```
Player 1 pieces = 29
Player 2 pieces = 22
Number of flipped pieces:  4
Total number of pieces on the board:  51
```



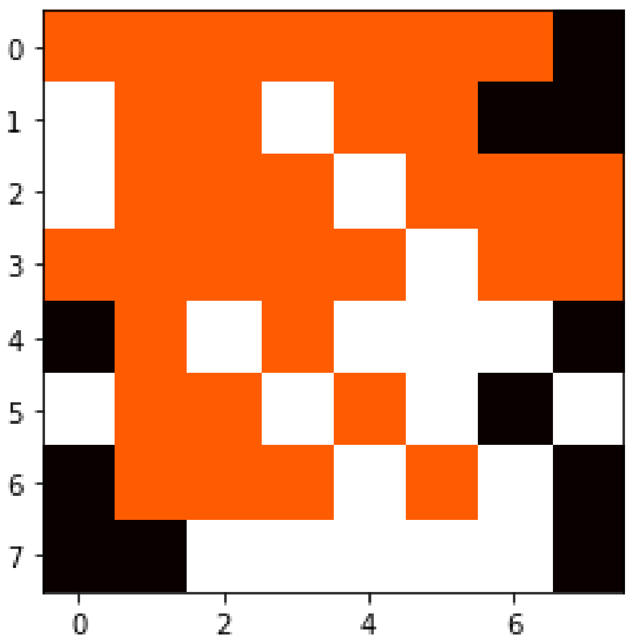
```
[50]: insert(5,7,2)
```

```
Player 1 pieces = 28
Player 2 pieces = 24
Number of flipped pieces:  1
Total number of pieces on the board:  52
```



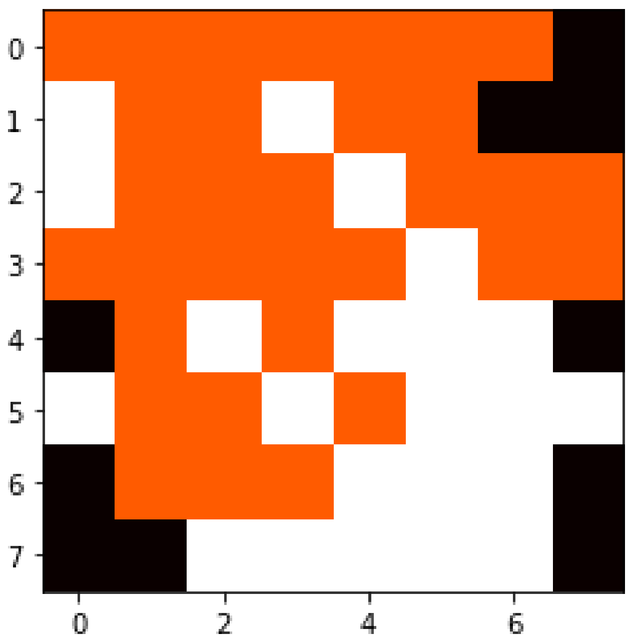
```
[51]: insert(2,1,1)
```

```
Player 1 pieces = 33
Player 2 pieces = 20
Number of flipped pieces:  4
Total number of pieces on the board:  53
```



```
[52]: insert(5,6,2)
```

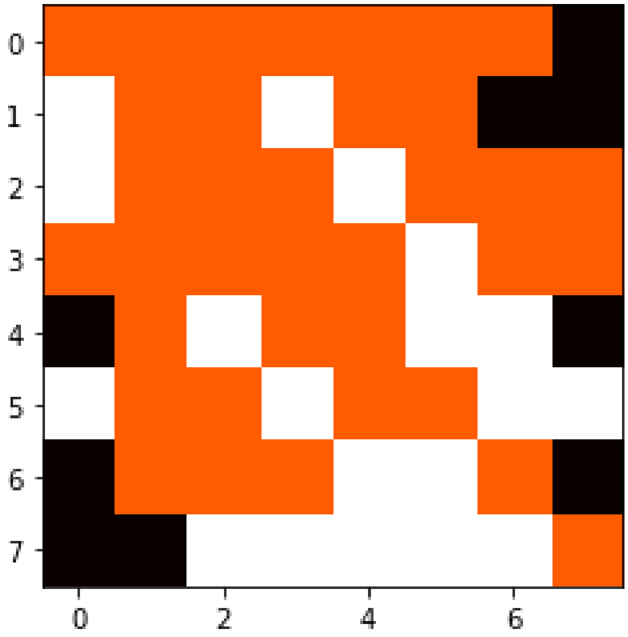
```
Player 1 pieces = 32
Player 2 pieces = 22
Number of flipped pieces:  1
Total number of pieces on the board:  54
```



[53]:

```
insert(7,7,1)
```

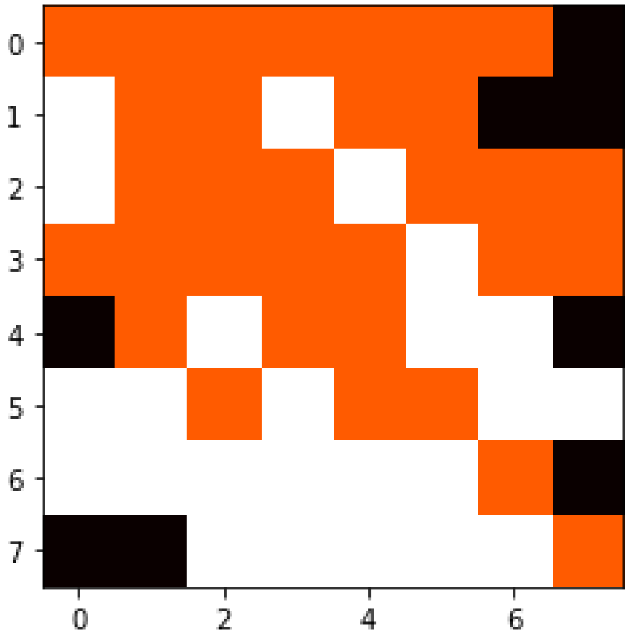
```
Player 1 pieces = 36
Player 2 pieces = 19
Number of flipped pieces:  3
Total number of pieces on the board:  55
```



[54]:

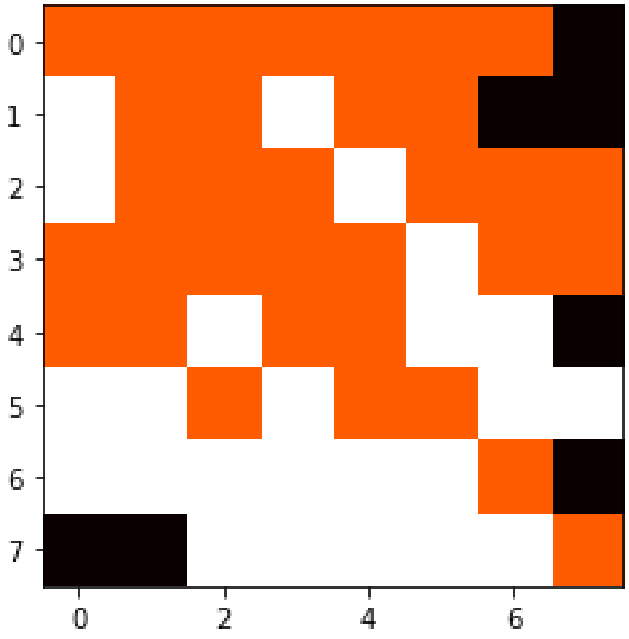
```
insert(6,0,2)
```

```
Player 1 pieces = 32
Player 2 pieces = 24
Number of flipped pieces:  4
Total number of pieces on the board:  56
```



```
[55]: insert(4,0,1)
```

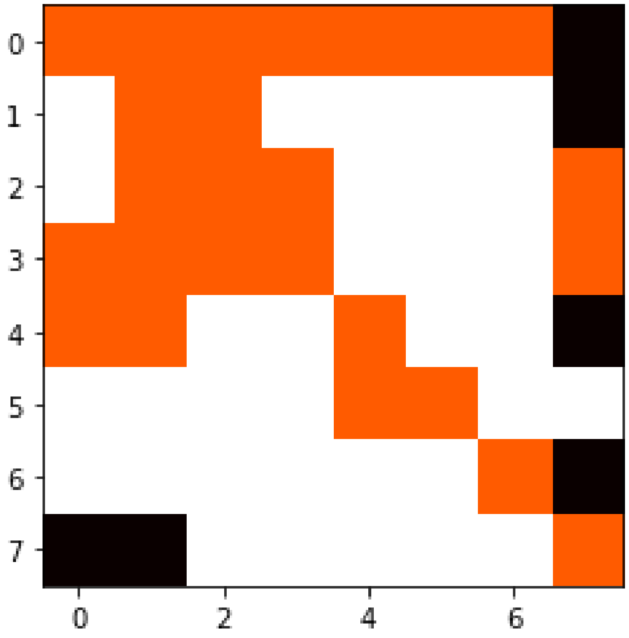
```
Player 1 pieces = 33
Player 2 pieces = 24
Number of flipped pieces:  0
Total number of pieces on the board:  57
```



[56]:

```
insert(1,6,2)
```

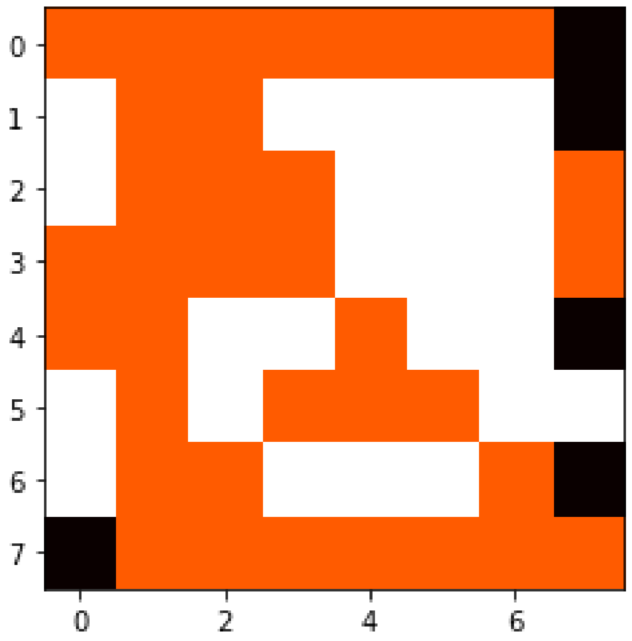
```
Player 1 pieces = 25
Player 2 pieces = 33
Number of flipped pieces:  8
Total number of pieces on the board:  58
```



[57]:

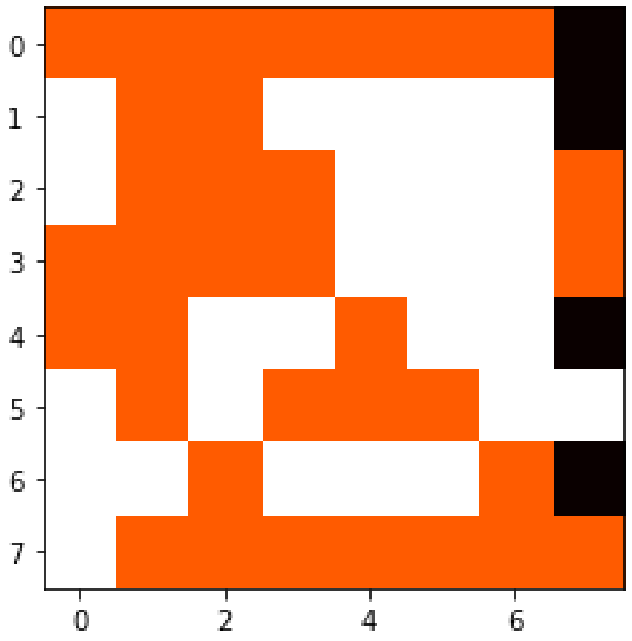
```
insert(7,1,1)
```

```
Player 1 pieces = 35
Player 2 pieces = 24
Number of flipped pieces:  9
Total number of pieces on the board:  59
```



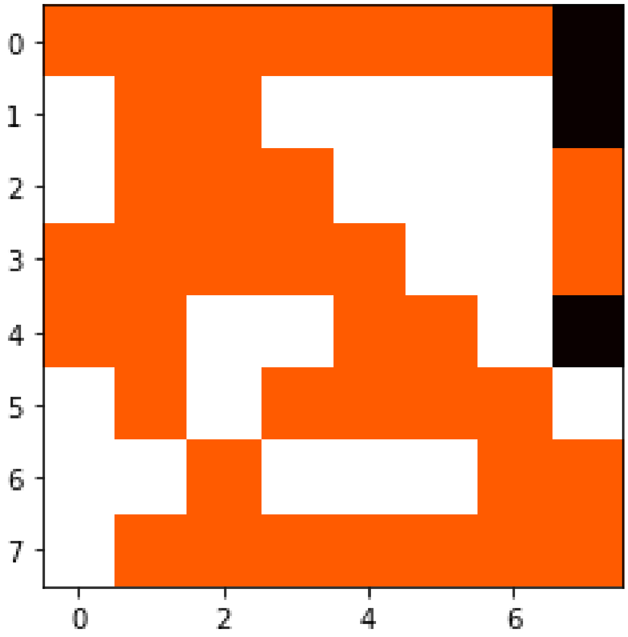
```
[58]: insert(7,0,2)
```

```
Player 1 pieces = 34
Player 2 pieces = 26
Number of flipped pieces:  1
Total number of pieces on the board:  60
```



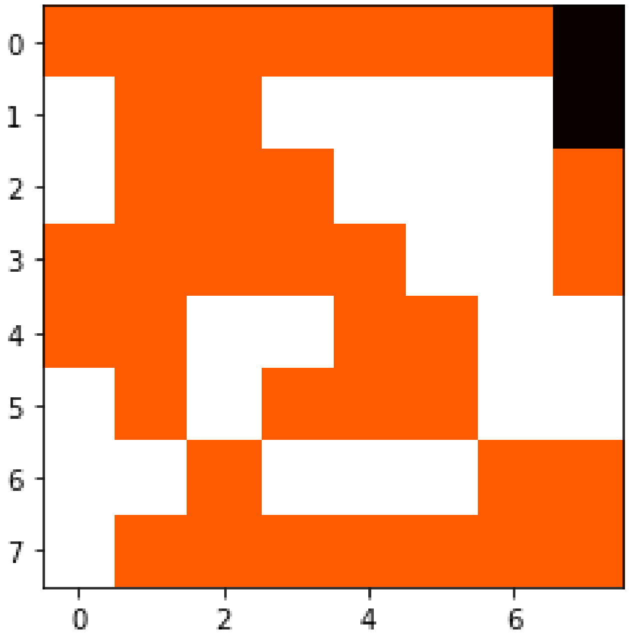
```
[59]: insert(6,7,1)
```

```
Player 1 pieces = 38
Player 2 pieces = 23
Number of flipped pieces:  3
Total number of pieces on the board:  61
```



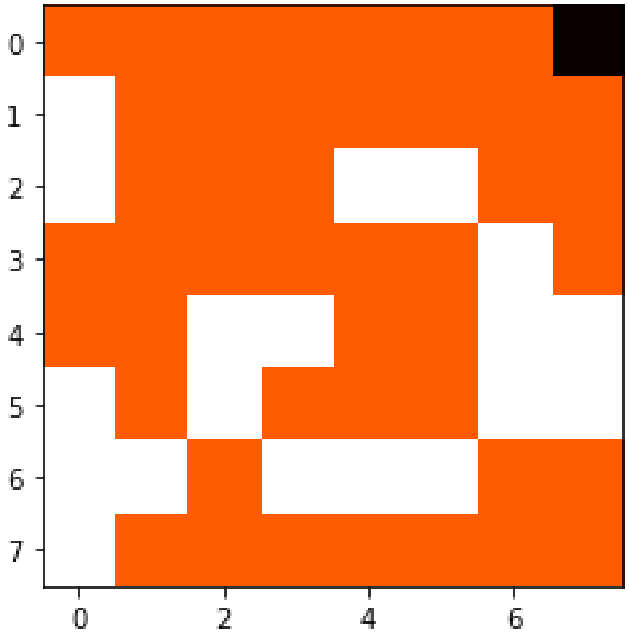
```
[60]: insert(4,7,2)
```

```
Player 1 pieces = 37
Player 2 pieces = 25
Number of flipped pieces:  1
Total number of pieces on the board:  62
```



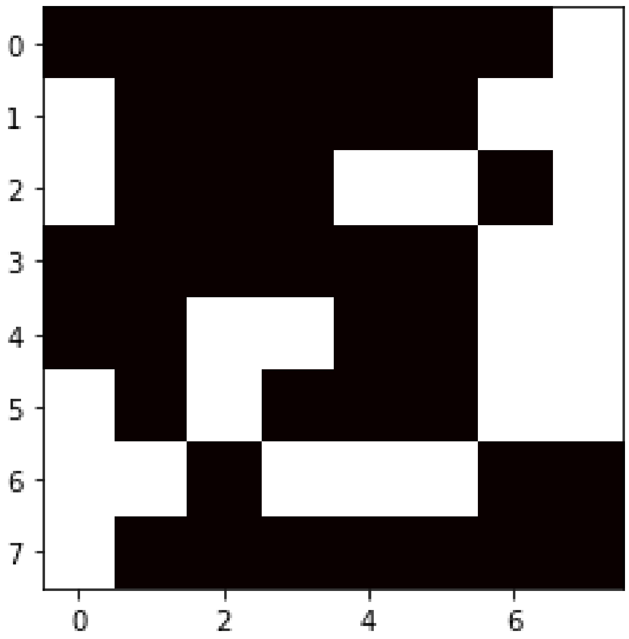
[61]:
```
insert(1,7,1)
```

```
Player 1 pieces = 44
Player 2 pieces = 19
Number of flipped pieces:  6
Total number of pieces on the board:  63
```



```
[62]: insert(0,7,2)
```

```
Player 1 pieces = 40
Player 2 pieces = 24
Number of flipped pieces:  4
Total number of pieces on the board:  64
```



```
[63]: print("Player 1 score =", np.count_nonzero(board==1))
      print("Player 2 score =", np.count_nonzero(board==2))
```

```
Player 1 score = 40
Player 2 score = 24
```

```
[ ]:
```