Leinster Water Polo - Fixtures Builder

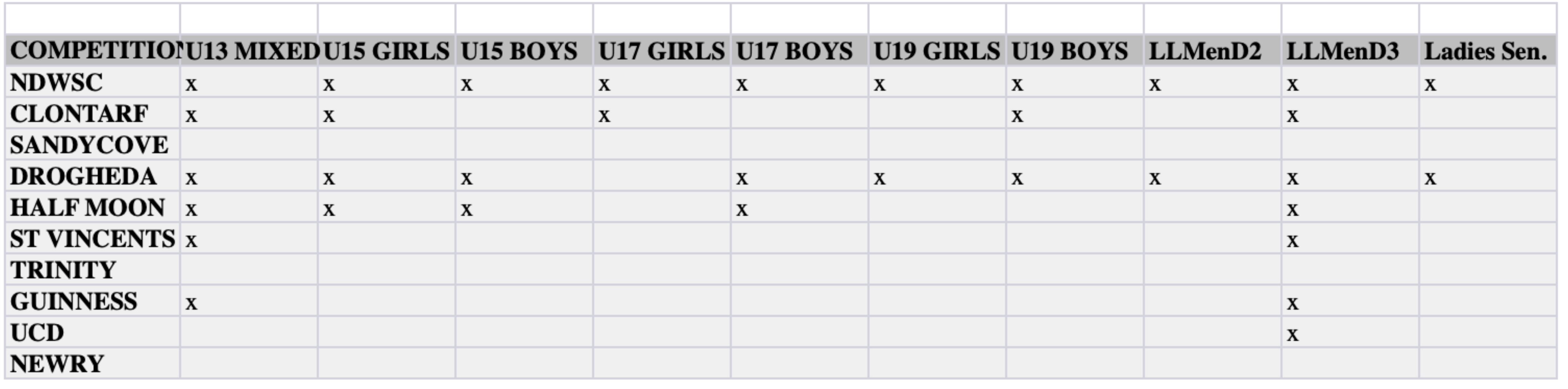
An overview of the Python code behind turning club competition entries in to a fixtures list.

By Duncan Mullaney - October 2021

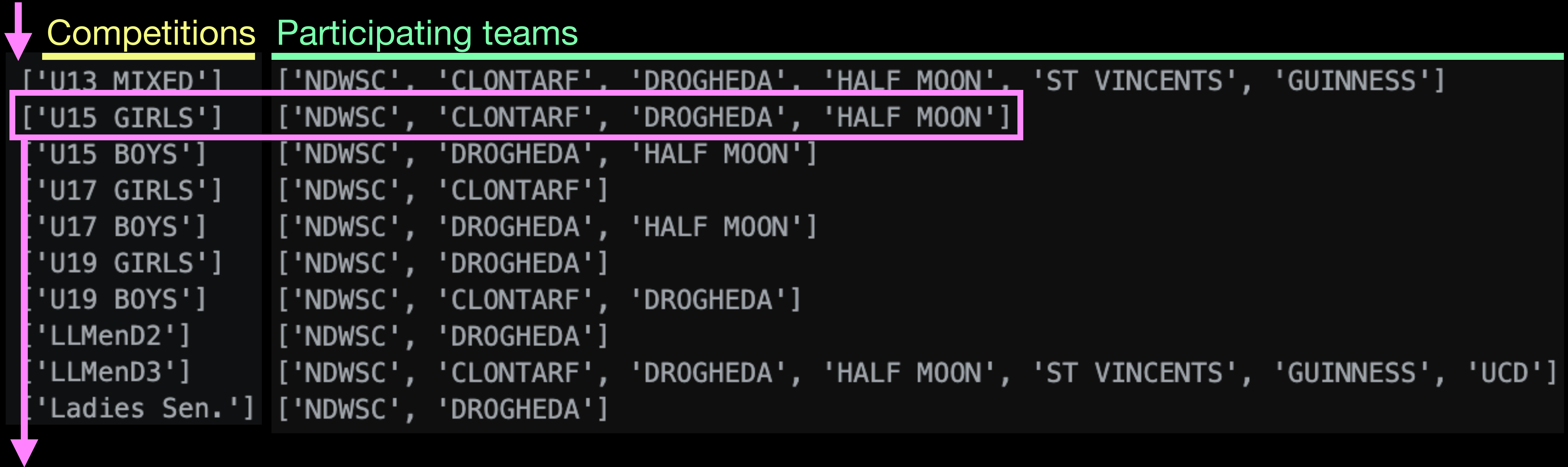
Table of Contents

1. Process Overview
2. Input (water polo clubs' entries)
3. Polygon Analysis
4. Play Matrix (PM)
5. PM Processing & Shifting
6. Shift Matrix
7. Print to Excel file
8. Code
9. Conclusion

1. Process Overview



COMPETITIO	U13 MIXED	U15 GIRLS	U15 BOYS	U17 GIRLS	U17 BOYS	U19 GIRLS	U19 BOYS	LLMenD2	LLMenD3	Ladies Sen.
NDWSC	x	x	x	x	x	x	x	x	x	x
CLONTARF	x	x		x			x		x	
SANDYCOVE										
DROGHEDA	x	x	x		x	x	x	x	x	x
HALF MOON	x	x	x		x				x	
ST VINCENTS	x								x	
TRINITY										
GUINNESS	x								x	
UCD									x	
NEWRY										

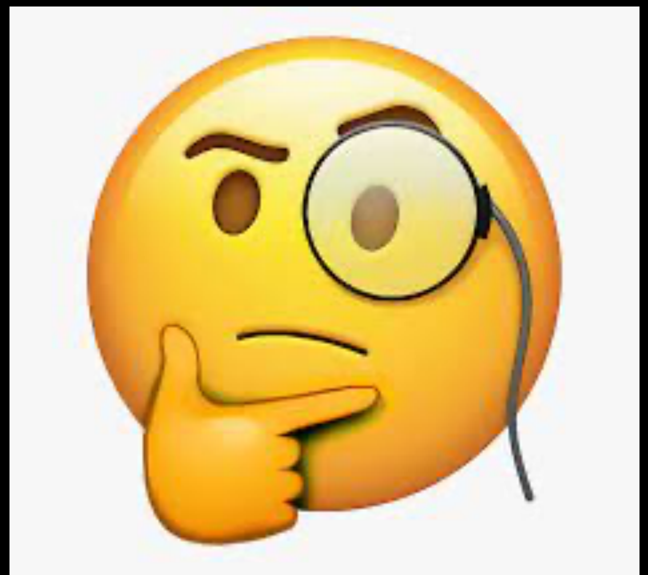


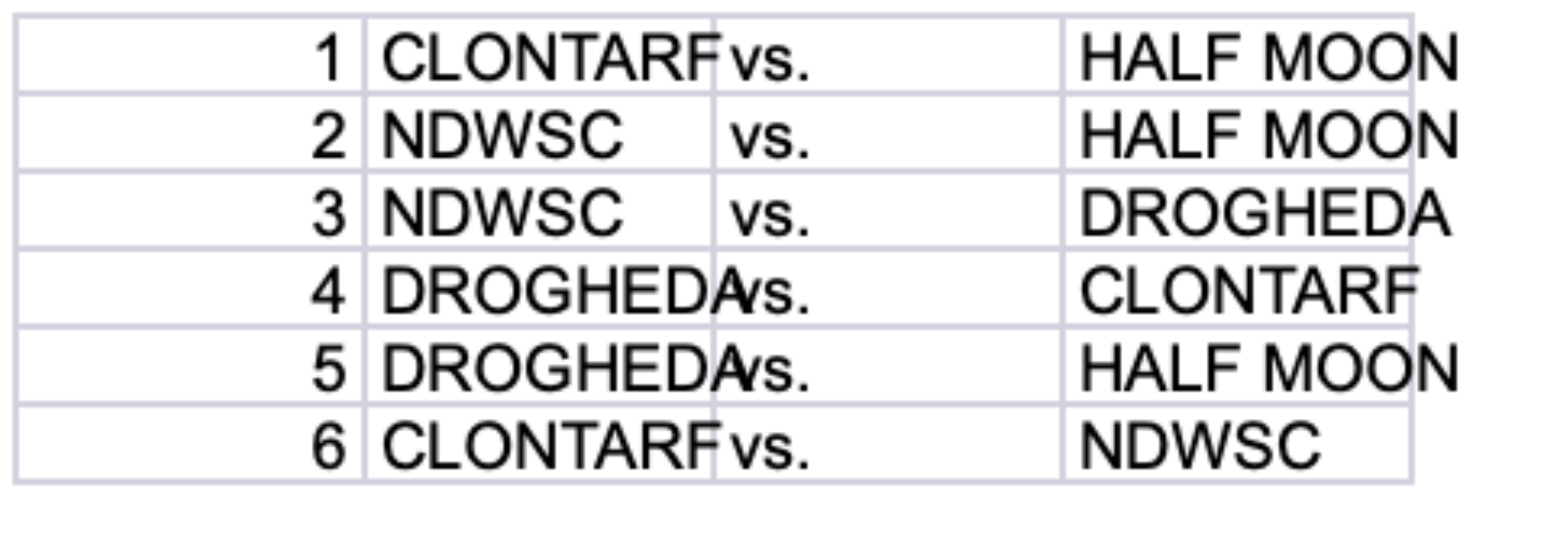
Competitions Participating teams

```
['U13 MIXED']   ['NDWSC', 'CLONTARF', 'DROGHEDA', 'HALF MOON', 'ST VINCENTS', 'GUINNESS']
['U15 GIRLS']   ['NDWSC', 'CLONTARF', 'DROGHEDA', 'HALF MOON']
['U15 BOYS']    ['NDWSC', 'DROGHEDA', 'HALF MOON']
['U17 GIRLS']   ['NDWSC', 'CLONTARF']
['U17 BOYS']    ['NDWSC', 'DROGHEDA', 'HALF MOON']
['U19 GIRLS']   ['NDWSC', 'DROGHEDA']
['U19 BOYS']    ['NDWSC', 'CLONTARF', 'DROGHEDA']
['LLMenD2']     ['NDWSC', 'DROGHEDA']
['LLMenD3']     ['NDWSC', 'CLONTARF', 'DROGHEDA', 'HALF MOON', 'ST VINCENTS', 'GUINNESS', 'UCD']
['Ladies Sen.'] ['NDWSC', 'DROGHEDA']
```

U15 Girls -> 4 teams -> 4C2 = 6 games

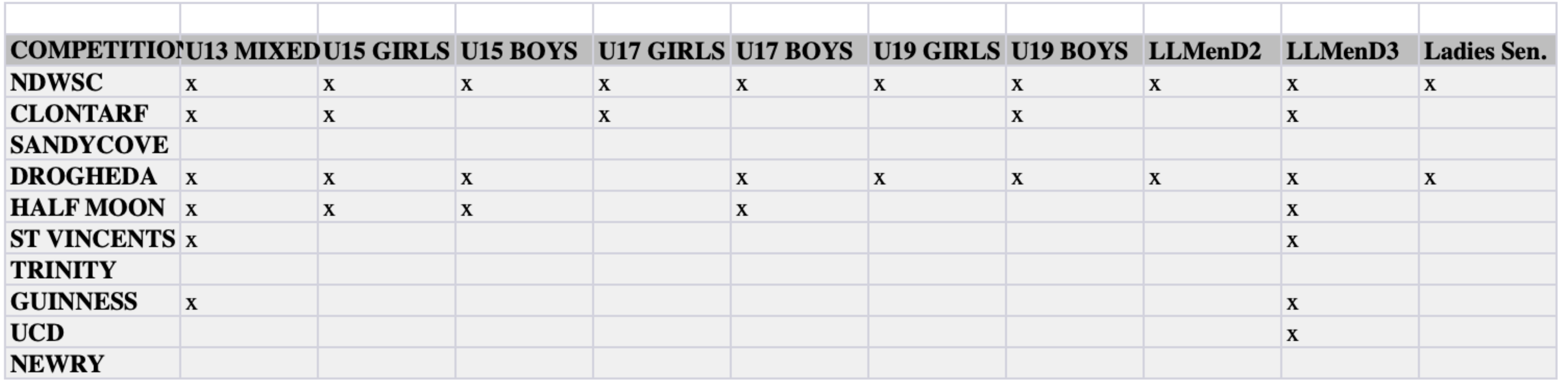
?





1	CLONTARF	vs.	HALF MOON
2	NDWSC	vs.	HALF MOON
3	NDWSC	vs.	DROGHEDA
4	DROGHEDA	vs.	CLONTARF
5	DROGHEDA	vs.	HALF MOON
6	CLONTARF	vs.	NDWSC

2. Input (water polo clubs' entries)



COMPETITIO	U13 MIXED	U15 GIRLS	U15 BOYS	U17 GIRLS	U17 BOYS	U19 GIRLS	U19 BOYS	LLMenD2	LLMenD3	Ladies Sen.
NDWSC	x	x	x	x	x	x	x	x	x	x
CLONTARF	x	x		x			x		x	
SANDYCOVE										
DROGHEDA	x	x	x		x	x	x	x	x	x
HALF MOON	x	x	x		x				x	
ST VINCENTS	x								x	
TRINITY										
GUINNESS	x								x	
UCD									x	
NEWRY										

3. Polygon Analysis

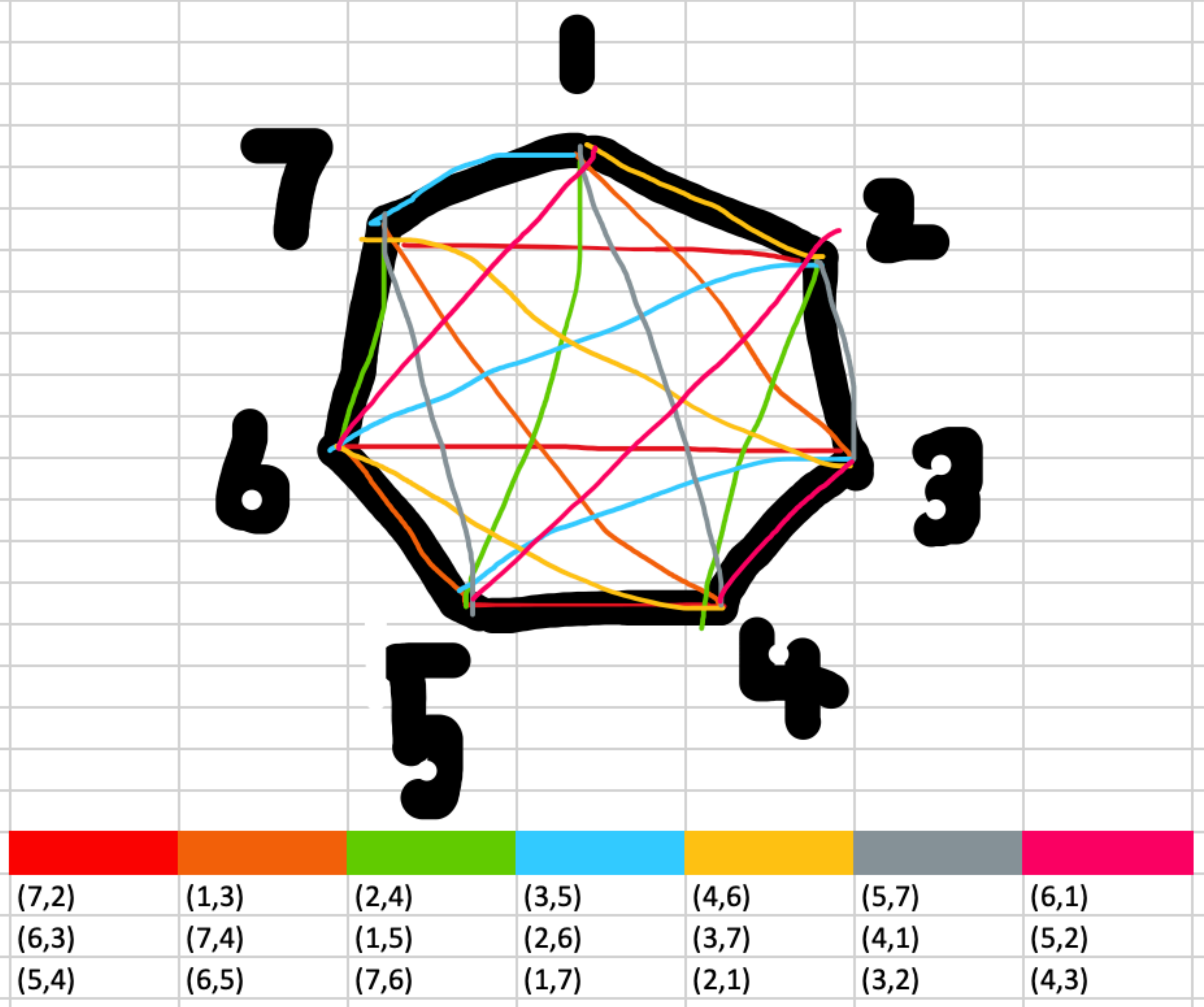
A “polygon” based method of generating a list of games in a round robin fashion was used (http://intermath.org/round-robin-tournament/).

Take the following example, for a competition with 7 teams participating, draw a polygon with one team at each vertex (1, 2, 3…).

Ignoring team 1, draw lines joining teams at opposite sides, e.g. join 7 and 2, 6 and 3, 5 and 4. This set of pairs makes up the first round of games.

Repeat this process, ignoring team 2 this time. The pairs will be (1,3), (7,4) and (6,5).

Repeat this for each vertex, until 7C2 pairings (i.e. 21) have been made (see table).



🟥	🟧	🟩	🟦	🟨	▨	🟪
(7,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	(6,1)
(6,3)	(7,4)	(1,5)	(2,6)	(3,7)	(4,1)	(5,2)
(5,4)	(6,5)	(7,6)	(1,7)	(2,1)	(3,2)	(4,3)

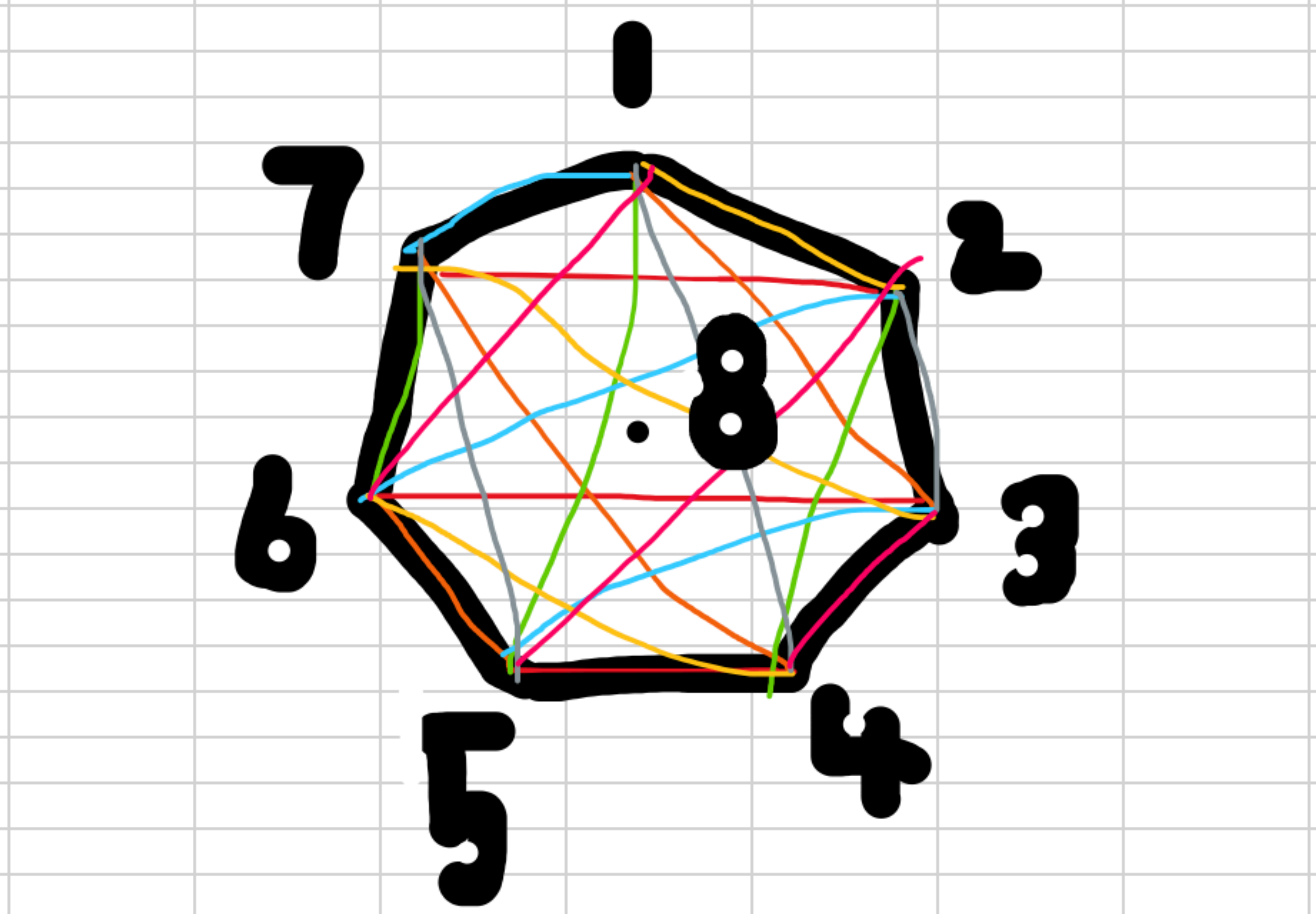
7.

3. Polygon Analysis

Similarly for a competition with 8 teams, draw the same polygon but with the eighth team in the centre.

Repeat this process, but instead of ignoring a team each time, pair this team with 8, such that the final pair will be (1,8) or (2,8) or (3,8) etc…

Repeat this for each vertex, until 8C2 pairings (i.e. 28) have been made.



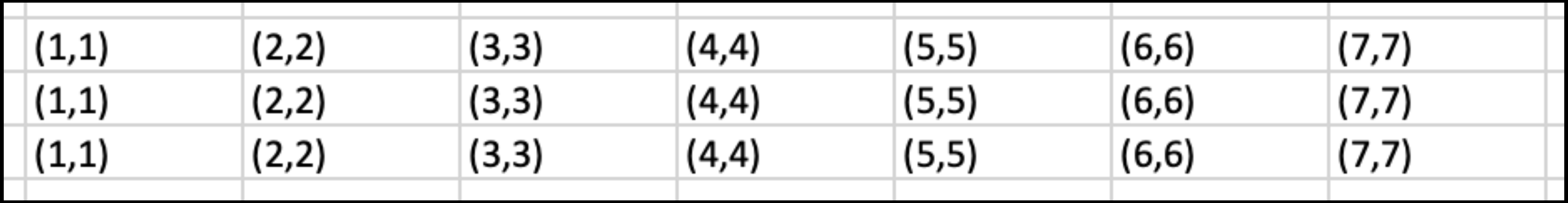
🟥	🟧	🟩	🟦	🟨	⬜	🟪
(7,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	(6,1)
(6,3)	(7,4)	(1,5)	(2,6)	(3,7)	(4,1)	(5,2)
(5,4)	(6,5)	(7,6)	(1,7)	(2,1)	(3,2)	(4,3)
(1,8)	(2,8)	(3,8)	(4,8)	(5,8)	(6,8)	(7,8)

8.

4. Play Matrix

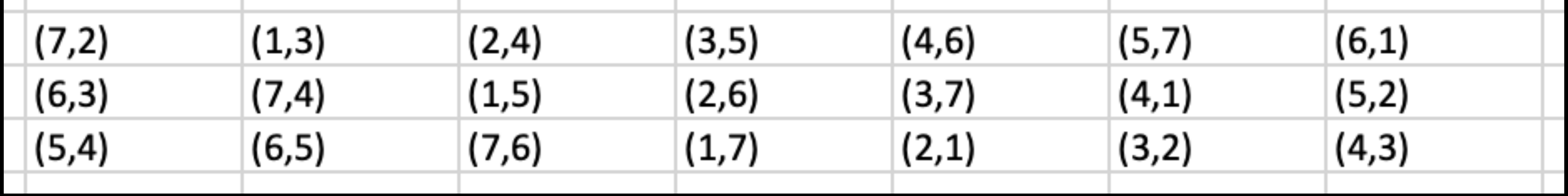
So the challenge is, with computer code, turn A in to B.

A



(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)

B



(7,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	(6,1)
(6,3)	(7,4)	(1,5)	(2,6)	(3,7)	(4,1)	(5,2)
(5,4)	(6,5)	(7,6)	(1,7)	(2,1)	(3,2)	(4,3)

This is called the “Play Matrix”

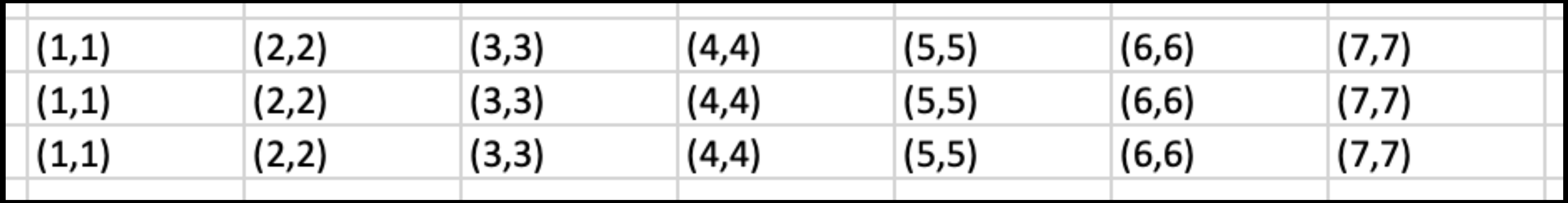
5. PM Processing & Shifting

You will notice the following in the play matrix:

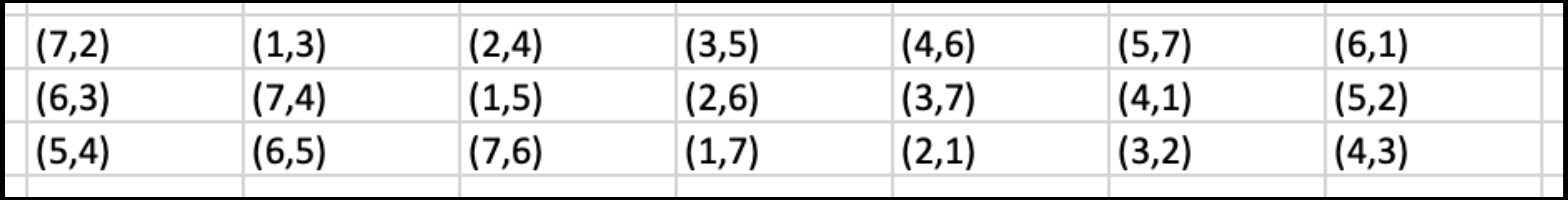
- in row 1, the first coordinate has been shifted by 1 cell, and the second coordinate has been shifted by 6 (i.e. [1,6]).

- In row 2, the first coordinate has been shifted by 2 cells, and the second coordinate has been shifted by 5 (i.e. [2,5]).

- In row 3, the first coordinate has been shifted by 3 cells, and the second coordinate has been shifted by 4 (i.e. [3,4]).



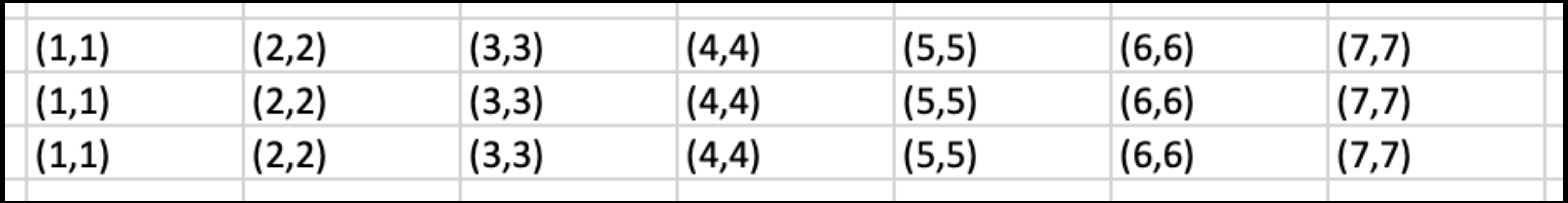
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)



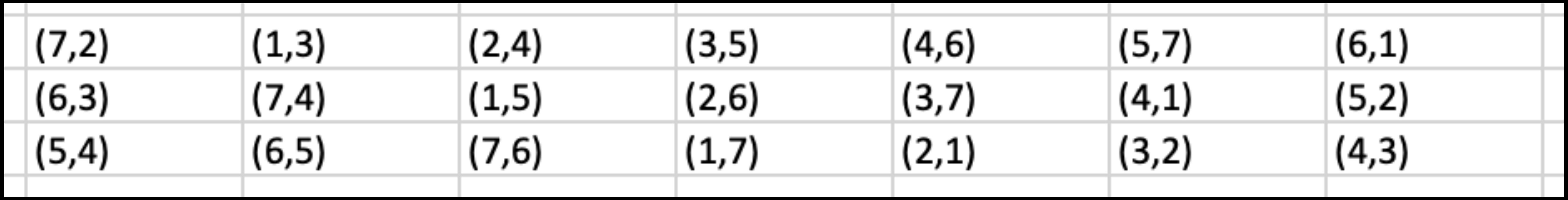
(7,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	(6,1)
(6,3)	(7,4)	(1,5)	(2,6)	(3,7)	(4,1)	(5,2)
(5,4)	(6,5)	(7,6)	(1,7)	(2,1)	(3,2)	(4,3)

…so there is a clear pattern here that can be utilised.

6. Shift Matrix



(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)

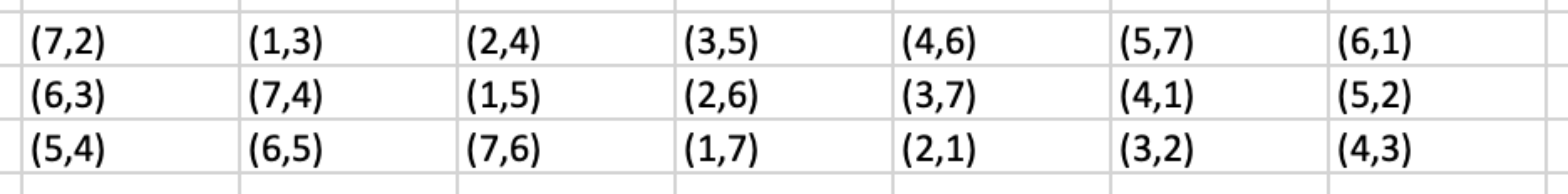


(7,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	(6,1)
(6,3)	(7,4)	(1,5)	(2,6)	(3,7)	(4,1)	(5,2)
(5,4)	(6,5)	(7,6)	(1,7)	(2,1)	(3,2)	(4,3)

This pattern gives rise to an associated "Shift Matrix"

… [[1,6] , [2,5] , [3,4]]

This matrix can be used as an instruction to shift the first coordinate in the first row by 1 and the second coordinate in the first row by 6, then the first coordinate in the second row by 2 and the second coordinate in the second row by 5, and so forth…

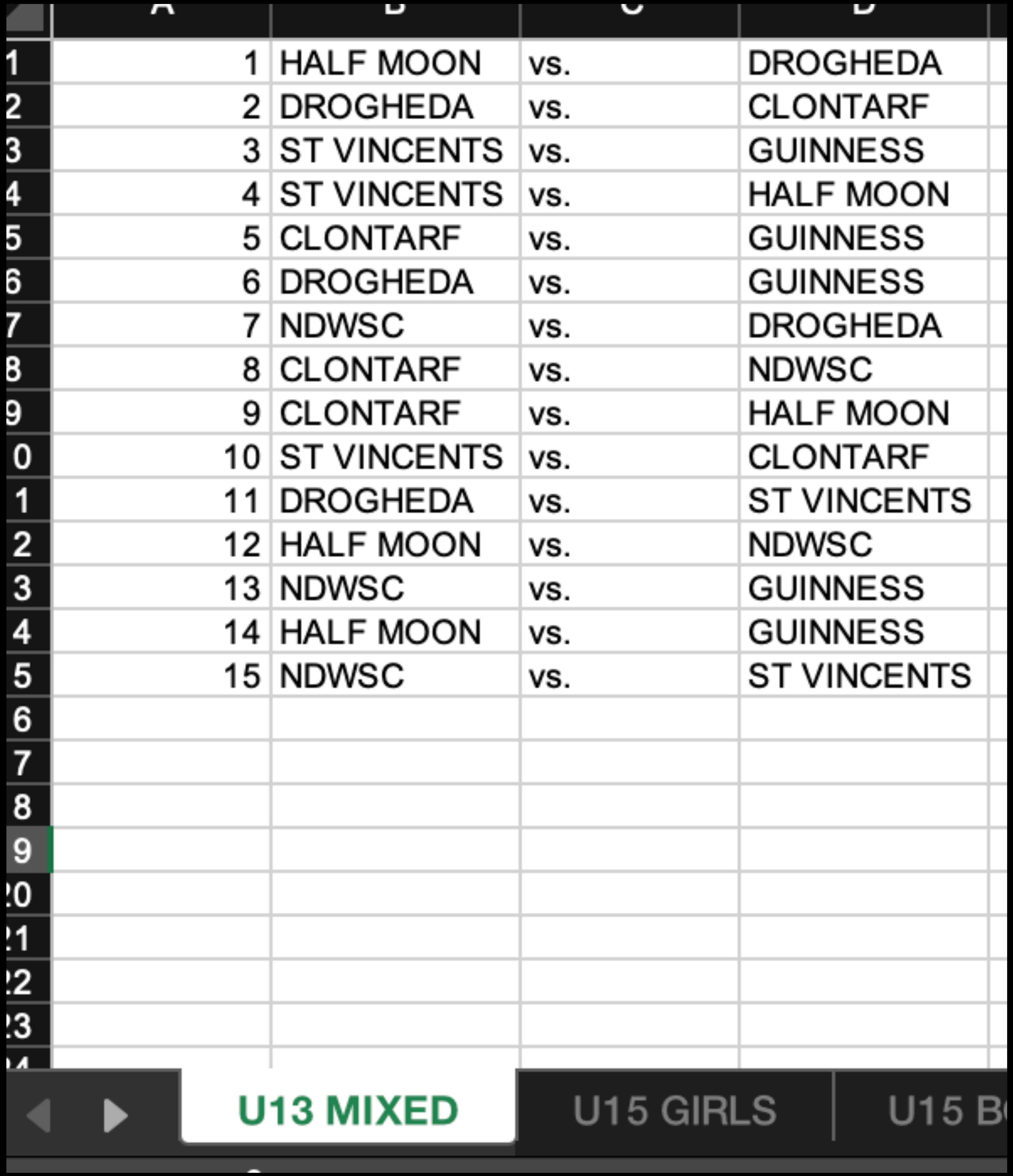


(7,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	(6,1)
(6,3)	(7,4)	(1,5)	(2,6)	(3,7)	(4,1)	(5,2)
(5,4)	(6,5)	(7,6)	(1,7)	(2,1)	(3,2)	(4,3)

7. Print to Excel file

Once the above play matrix format has been achieved, the next step is to use it to print a list of games like so to an excel file output

The code creates a new sheet for each competition (in this case "U13 MIXED"), and prints the team that each pairing corresponds to



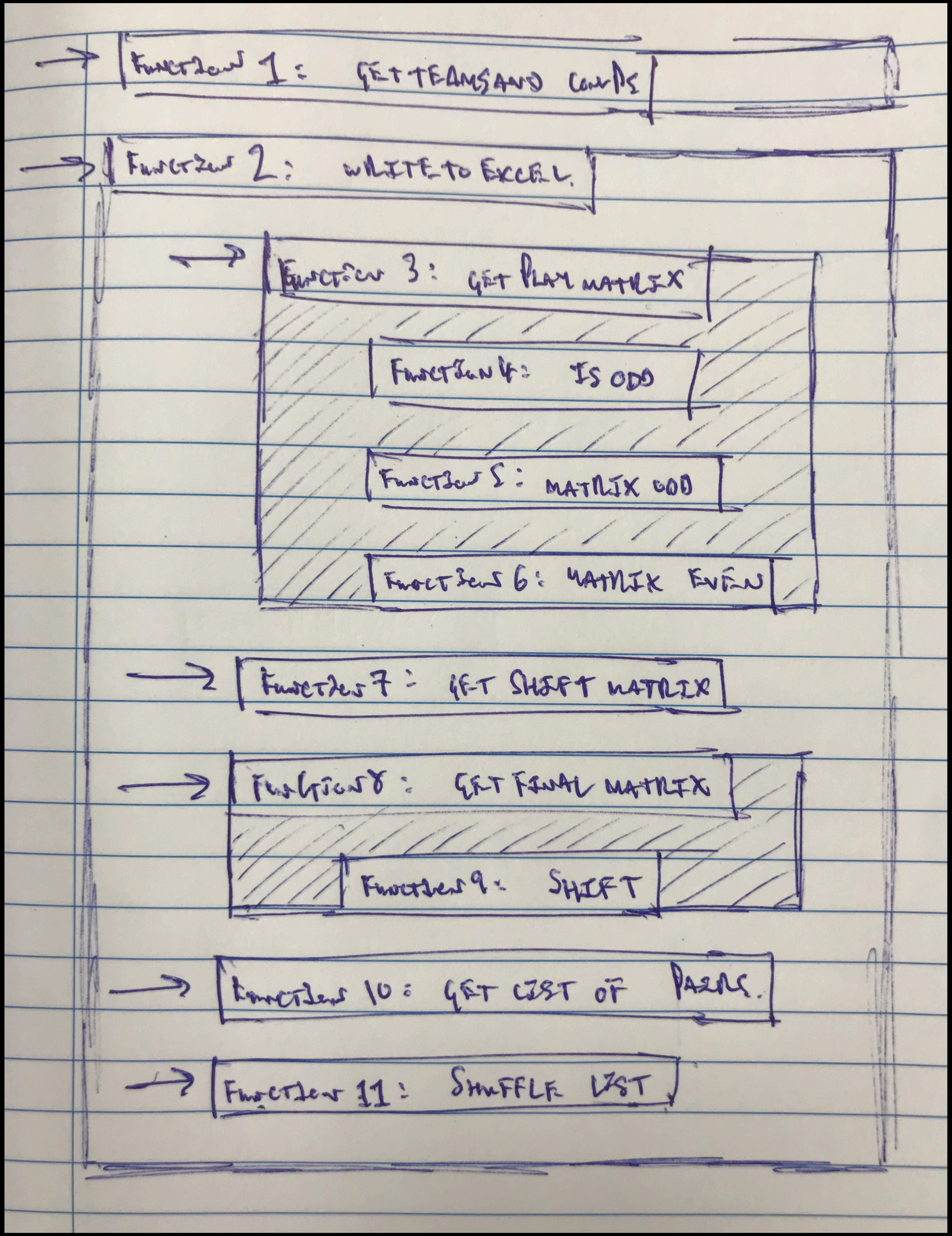
1	HALF MOON	vs.	DROGHEDA
2	DROGHEDA	vs.	CLONTARF
3	ST VINCENTS	vs.	GUINNESS
4	ST VINCENTS	vs.	HALF MOON
5	CLONTARF	vs.	GUINNESS
6	DROGHEDA	vs.	GUINNESS
7	NDWSC	vs.	DROGHEDA
8	CLONTARF	vs.	NDWSC
9	CLONTARF	vs.	HALF MOON
10	ST VINCENTS	vs.	CLONTARF
11	DROGHEDA	vs.	ST VINCENTS
12	HALF MOON	vs.	NDWSC
13	NDWSC	vs.	GUINNESS
14	HALF MOON	vs.	GUINNESS
15	NDWSC	vs.	ST VINCENTS

8. Code

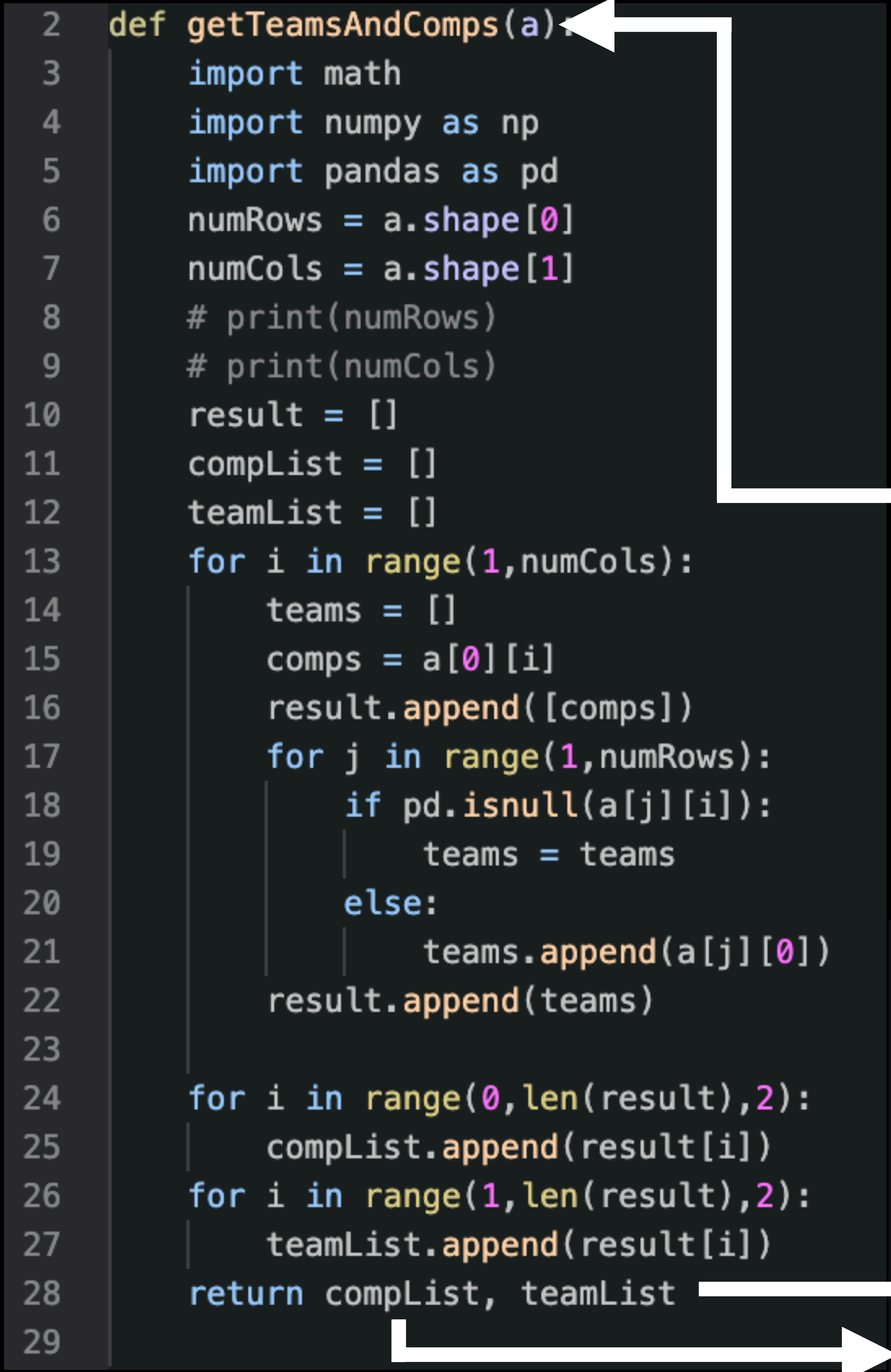
A total of 11 functions are defined within the code, and are used in the following structure.

Two primary functions are called, the second of which calls a number of sub-functions, and some of these utilise *more* sub-functions.

The purpose of these functions will be discussed on a high level in the following slides.


FUNCTION 1: GET TEAMS AND COMPS
FUNCTION 2: WRITE TO EXCEL.
FUNCTION 3: GET PLAY MATRIX
FUNCTION 4: IS ODD
FUNCTION 5: MATRIX ODD
FUNCTION 6: MATRIX EVEN
FUNCTION 7: GET SHIFT MATRIX
FUNCTION 8: GET FINAL MATRIX
FUNCTION 9: SHIFT
FUNCTION 10: GET LIST OF PAIRS.
FUNCTION 11: SHUFFLE LIST

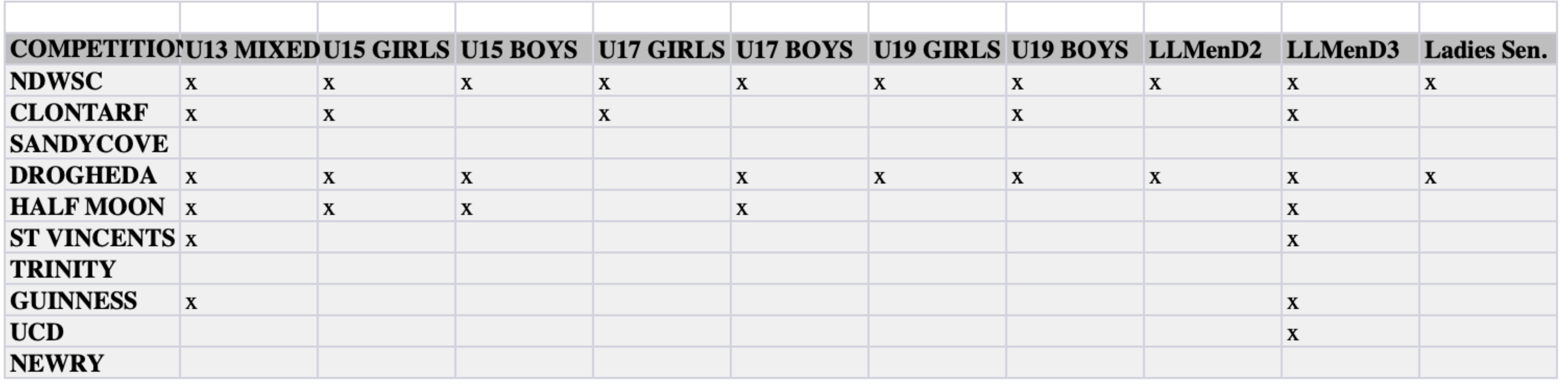
Function 1; “*getTeamAndComps*”



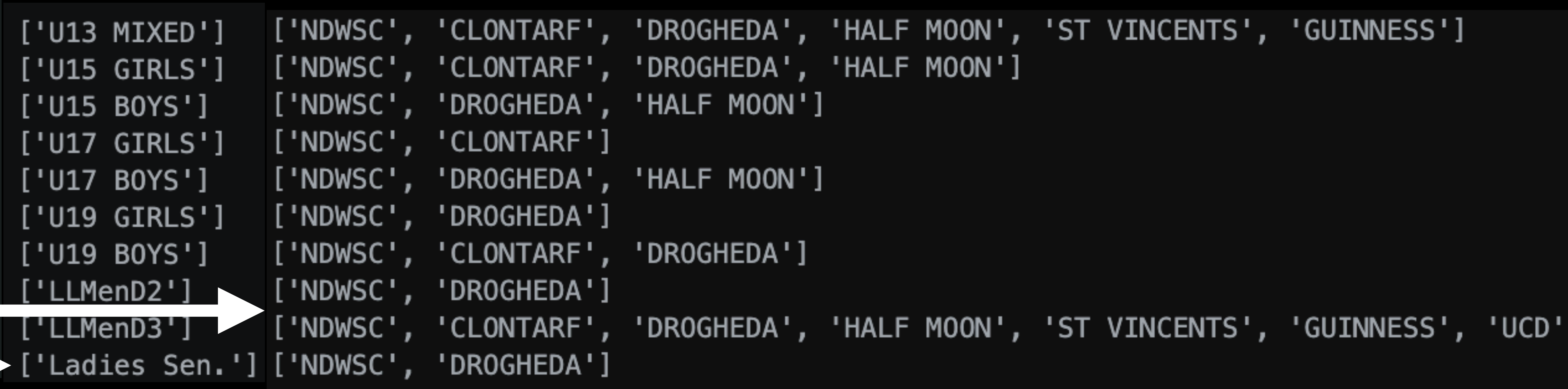
```
def getTeamsAndComps(a):
    import math
    import numpy as np
    import pandas as pd
    numRows = a.shape[0]
    numCols = a.shape[1]
    # print(numRows)
    # print(numCols)
    result = []
    compList = []
    teamList = []
    for i in range(1,numCols):
        teams = []
        comps = a[0][i]
        result.append([comps])
        for j in range(1,numRows):
            if pd.isnull(a[j][i]):
                teams = teams
            else:
                teams.append(a[j][0])
        result.append(teams)

    for i in range(0,len(result),2):
        compList.append(result[i])
    for i in range(1,len(result),2):
        teamList.append(result[i])
    return compList, teamList
```

This function takes in the grid of teams and the entries as below, interprets this an array, and returns a list of each competition and its respective participants as a corresponding list of teams.

This is *function 1*.

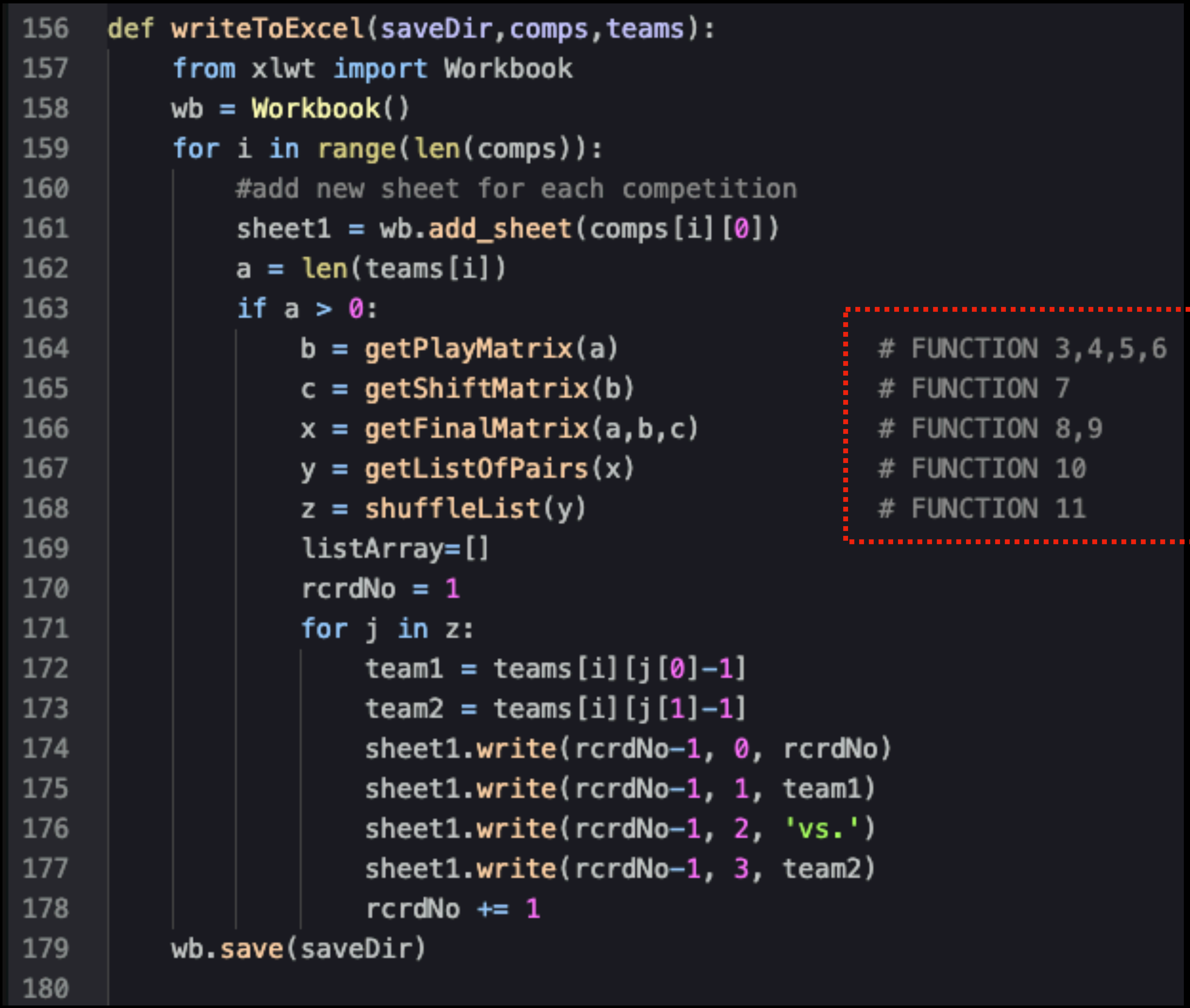


COMPETITIO	U13 MIXED	U15 GIRLS	U15 BOYS	U17 GIRLS	U17 BOYS	U19 GIRLS	U19 BOYS	LLMenD2	LLMenD3	Ladies Sen.
NDWSC	x	x	x	x	x	x	x	x	x	x
CLONTARF	x	x		x			x		x	
SANDYCOVE										
DROGHEDA	x	x	x		x	x	x	x	x	x
HALF MOON	x	x	x		x				x	
ST VINCENTS	x								x	
TRINITY										
GUINNESS	x								x	
UCD									x	
NEWRY										



```
['U13 MIXED']   ['NDWSC', 'CLONTARF', 'DROGHEDA', 'HALF MOON', 'ST VINCENTS', 'GUINNESS']
['U15 GIRLS']   ['NDWSC', 'CLONTARF', 'DROGHEDA', 'HALF MOON']
['U15 BOYS']    ['NDWSC', 'DROGHEDA', 'HALF MOON']
['U17 GIRLS']   ['NDWSC', 'CLONTARF']
['U17 BOYS']    ['NDWSC', 'DROGHEDA', 'HALF MOON']
['U19 GIRLS']   ['NDWSC', 'DROGHEDA']
['U19 BOYS']    ['NDWSC', 'CLONTARF', 'DROGHEDA']
['LLMenD2']     ['NDWSC', 'DROGHEDA']
['LLMenD3']     ['NDWSC', 'CLONTARF', 'DROGHEDA', 'HALF MOON', 'ST VINCENTS', 'GUINNESS', 'UCD'
['Ladies Sen.'] ['NDWSC', 'DROGHEDA']
```

Function 2; “*writeToExcel*”

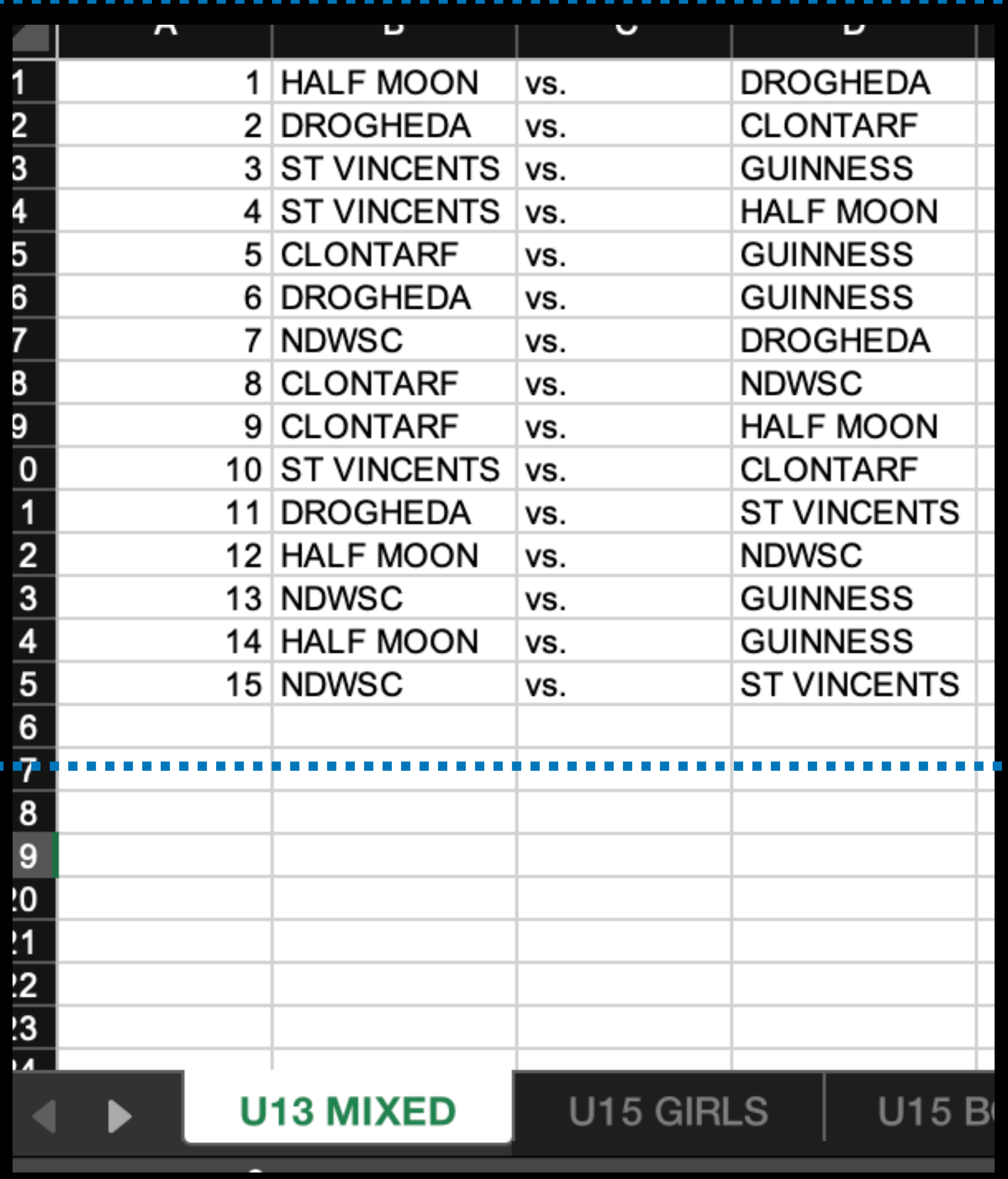


```python
def writeToExcel(saveDir,comps,teams):
    from xlwt import Workbook
    wb = Workbook()
    for i in range(len(comps)):
        #add new sheet for each competition
        sheet1 = wb.add_sheet(comps[i][0])
        a = len(teams[i])
        if a > 0:
            b = getPlayMatrix(a)                    # FUNCTION 3,4,5,6
            c = getShiftMatrix(b)                   # FUNCTION 7
            x = getFinalMatrix(a,b,c)               # FUNCTION 8,9
            y = getListOfPairs(x)                   # FUNCTION 10
            z = shuffleList(y)                      # FUNCTION 11
            listArray=[]
            rcrdNo = 1
            for j in z:
                team1 = teams[i][j[0]-1]
                team2 = teams[i][j[1]-1]
                sheet1.write(rcrdNo-1, 0, rcrdNo)
                sheet1.write(rcrdNo-1, 1, team1)
                sheet1.write(rcrdNo-1, 2, 'vs.')
                sheet1.write(rcrdNo-1, 3, team2)
                rcrdNo += 1
    wb.save(saveDir)

```

This function takes in the list of each competition and teams, cycles through each competition and, using functions 3-11 (which are discussed in subsequent slides), generates the list of match-ups required for each competition, i.e. the goal of the project.

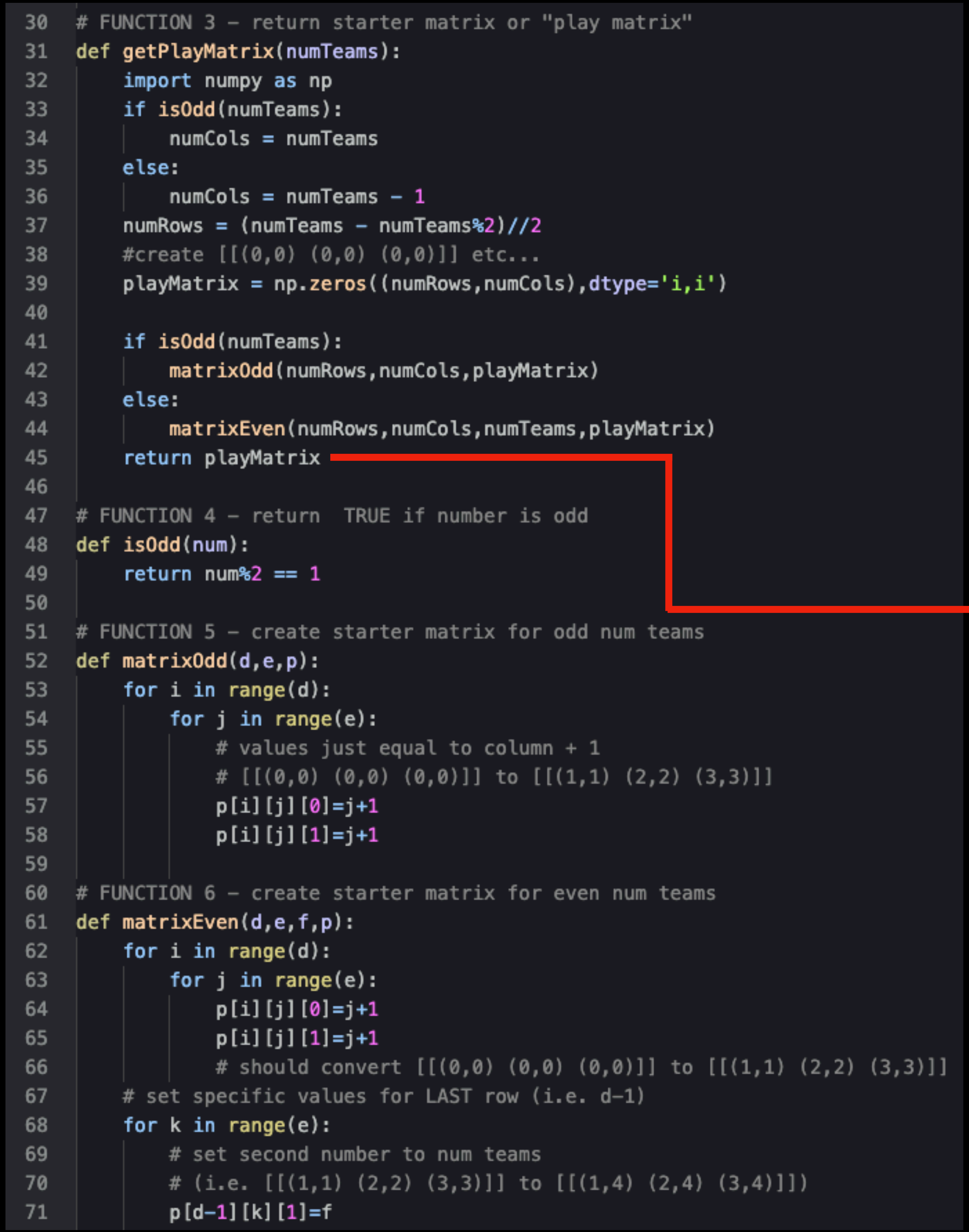
This is *function 2*.

See below a sample output for the U13 mixed competition;

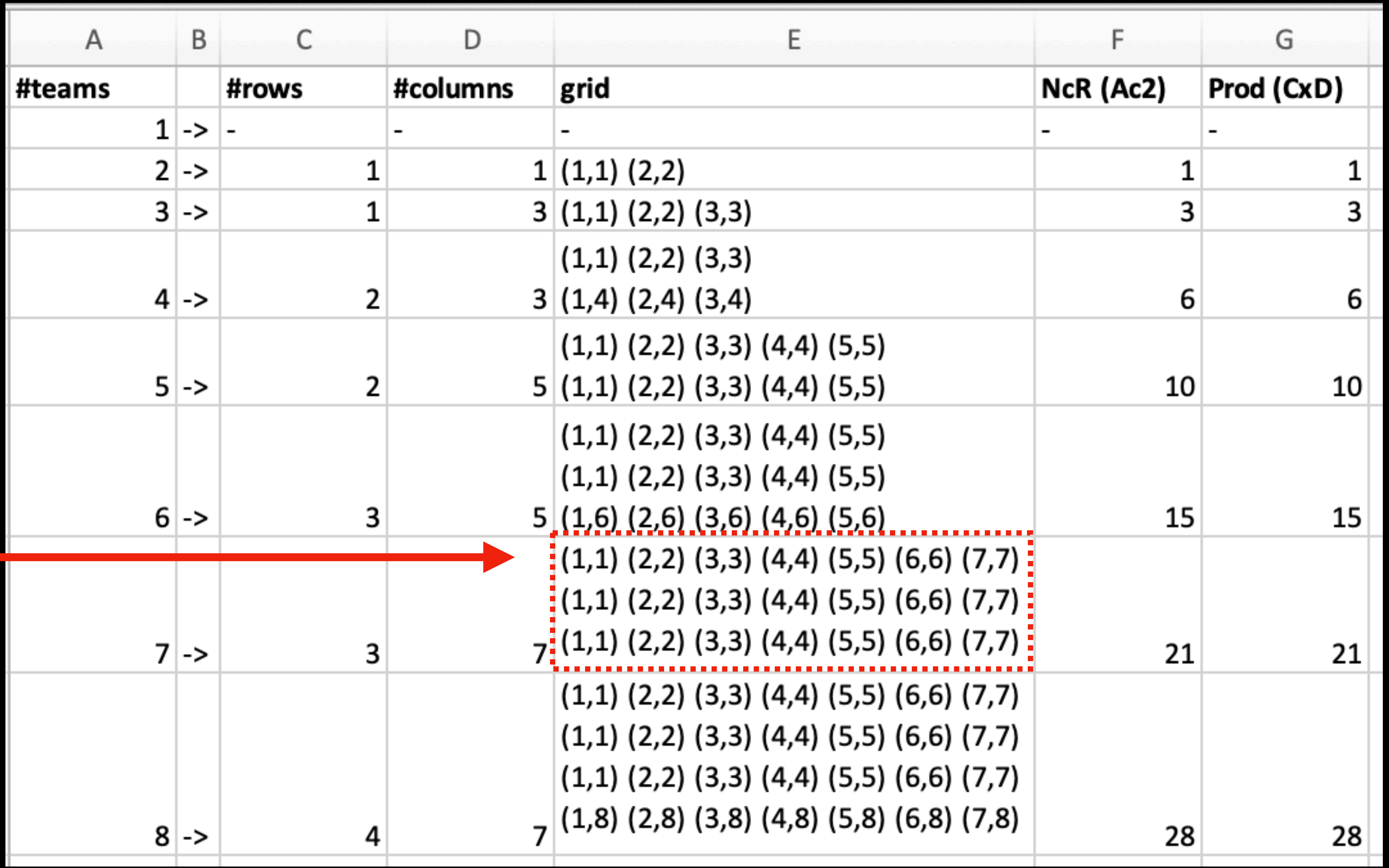


A	B	C	D
1	HALF MOON	vs.	DROGHEDA
2	DROGHEDA	vs.	CLONTARF
3	ST VINCENTS	vs.	GUINNESS
4	ST VINCENTS	vs.	HALF MOON
5	CLONTARF	vs.	GUINNESS
6	DROGHEDA	vs.	GUINNESS
7	NDWSC	vs.	DROGHEDA
8	CLONTARF	vs.	NDWSC
9	CLONTARF	vs.	HALF MOON
10	ST VINCENTS	vs.	CLONTARF
11	DROGHEDA	vs.	ST VINCENTS
12	HALF MOON	vs.	NDWSC
13	NDWSC	vs.	GUINNESS
14	HALF MOON	vs.	GUINNESS
15	NDWSC	vs.	ST VINCENTS

Function 3; “*getPlayMatrix*”



```
# FUNCTION 3 - return starter matrix or "play matrix"
def getPlayMatrix(numTeams):
    import numpy as np
    if isOdd(numTeams):
        numCols = numTeams
    else:
        numCols = numTeams - 1
    numRows = (numTeams - numTeams%2)//2
    #create [[(0,0) (0,0) (0,0)]] etc...
    playMatrix = np.zeros((numRows,numCols),dtype='i,i')

    if isOdd(numTeams):
        matrixOdd(numRows,numCols,playMatrix)
    else:
        matrixEven(numRows,numCols,numTeams,playMatrix)
    return playMatrix

# FUNCTION 4 - return  TRUE if number is odd
def isOdd(num):
    return num%2 == 1

# FUNCTION 5 - create starter matrix for odd num teams
def matrixOdd(d,e,p):
    for i in range(d):
        for j in range(e):
            # values just equal to column + 1
            # [[(0,0) (0,0) (0,0)]] to [[(1,1) (2,2) (3,3)]]
            p[i][j][0]=j+1
            p[i][j][1]=j+1

# FUNCTION 6 - create starter matrix for even num teams
def matrixEven(d,e,f,p):
    for i in range(d):
        for j in range(e):
            p[i][j][0]=j+1
            p[i][j][1]=j+1
            # should convert [[(0,0) (0,0) (0,0)]] to [[(1,1) (2,2) (3,3)]]
    # set specific values for LAST row (i.e. d-1)
    for k in range(e):
        # set second number to num teams
        # (i.e. [[(1,1) (2,2) (3,3)]] to [[(1,4) (2,4) (3,4)]])
        p[d-1][k][1]=f
```

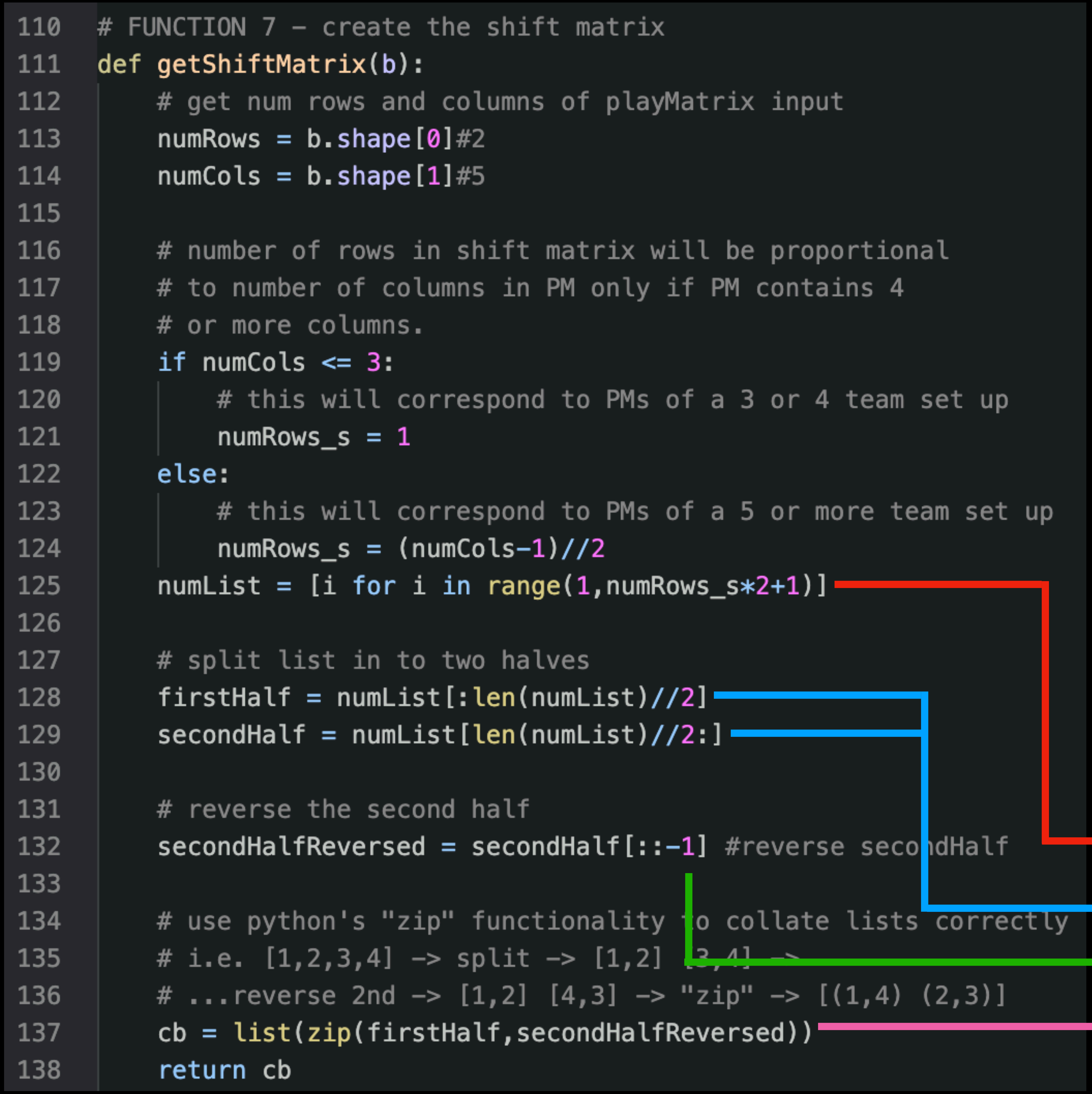


A	B	C	D	E	F	G
#teams		#rows	#columns	grid	NcR (Ac2)	Prod (CxD)
1	->	-	-	-	-	-
2	->	1	1	(1,1) (2,2)	1	1
3	->	1	3	(1,1) (2,2) (3,3)	3	3
4	->	2	3	(1,1) (2,2) (3,3) (1,4) (2,4) (3,4)	6	6
5	->	2	5	(1,1) (2,2) (3,3) (4,4) (5,5) (1,1) (2,2) (3,3) (4,4) (5,5)	10	10
6	->	3	5	(1,1) (2,2) (3,3) (4,4) (5,5) (1,1) (2,2) (3,3) (4,4) (5,5) (1,6) (2,6) (3,6) (4,6) (5,6)	15	15
7	->	3	7	(1,1) (2,2) (3,3) (4,4) (5,5) (6,6) (7,7) (1,1) (2,2) (3,3) (4,4) (5,5) (6,6) (7,7) (1,1) (2,2) (3,3) (4,4) (5,5) (6,6) (7,7)	21	21
8	->	4	7	(1,1) (2,2) (3,3) (4,4) (5,5) (6,6) (7,7) (1,1) (2,2) (3,3) (4,4) (5,5) (6,6) (7,7) (1,1) (2,2) (3,3) (4,4) (5,5) (6,6) (7,7) (1,8) (2,8) (3,8) (4,8) (5,8) (6,8) (7,8)	28	28

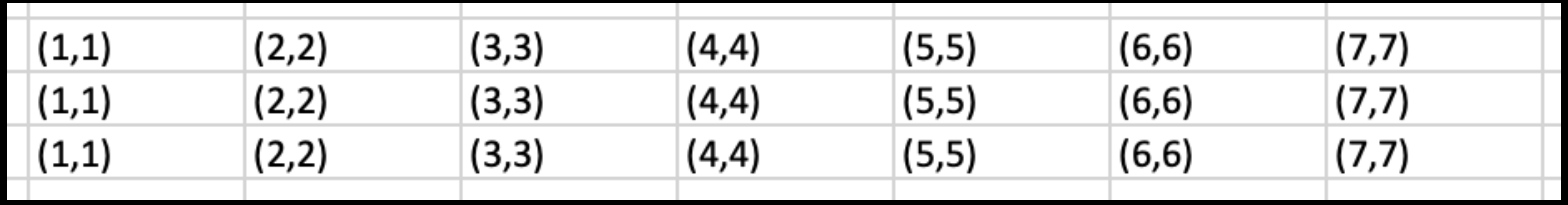
The code on the left produces what is referred to as a “play matrix”, which (at least initially) looks like the array of number pairs as see in column “E” I the excel screen-grab.

So where there are, say, 7 teams who entered in to a competition, the play matrix will begin as such in column “E”.

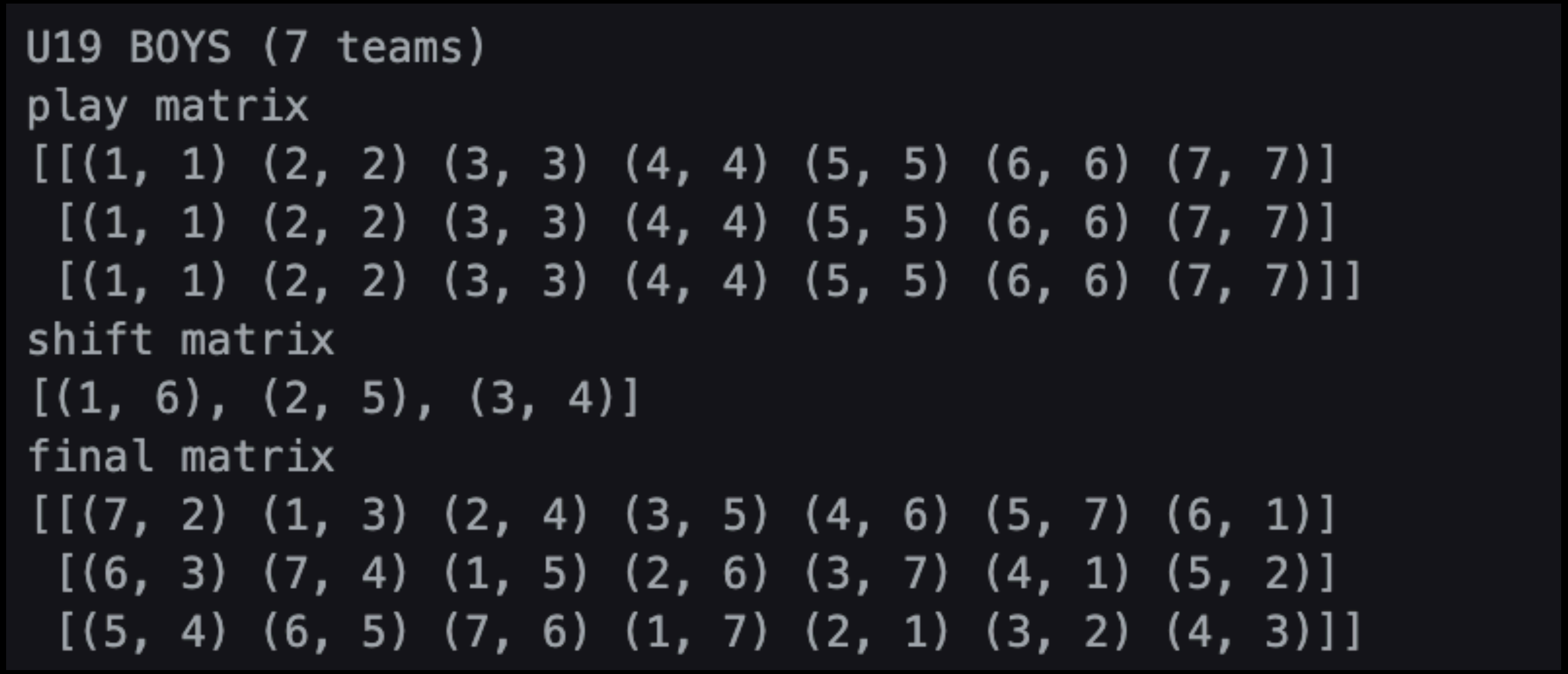
Function 7; “*getShiftMatrix*”



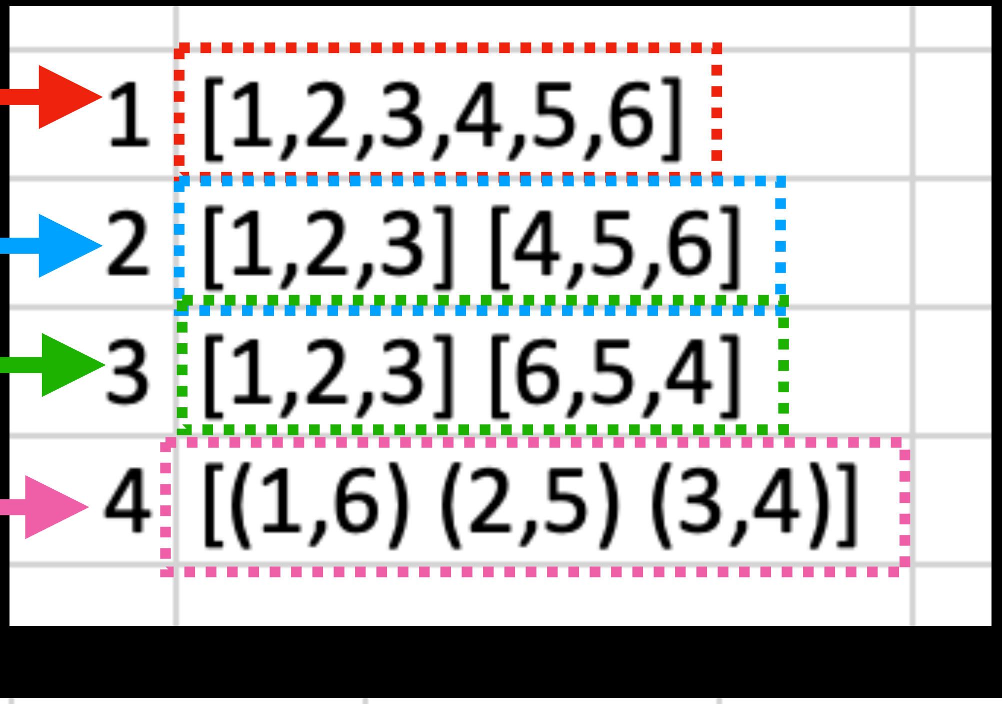
```python
# FUNCTION 7 - create the shift matrix
def getShiftMatrix(b):
    # get num rows and columns of playMatrix input
    numRows = b.shape[0]#2
    numCols = b.shape[1]#5

    # number of rows in shift matrix will be proportional
    # to number of columns in PM only if PM contains 4
    # or more columns.
    if numCols <= 3:
        # this will correspond to PMs of a 3 or 4 team set up
        numRows_s = 1
    else:
        # this will correspond to PMs of a 5 or more team set up
        numRows_s = (numCols-1)//2
    numList = [i for i in range(1,numRows_s*2+1)]

    # split list in to two halves
    firstHalf = numList[:len(numList)//2]
    secondHalf = numList[len(numList)//2:]

    # reverse the second half
    secondHalfReversed = secondHalf[::-1] #reverse secondHalf

    # use python's "zip" functionality to collate lists correctly
    # i.e. [1,2,3,4] -> split -> [1,2] [3,4] ->
    # ...reverse 2nd -> [1,2] [4,3] -> "zip" -> [(1,4) (2,3)]
    cb = list(zip(firstHalf,secondHalfReversed))
    return cb
```



(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)
(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)



```
U19 BOYS (7 teams)
play matrix
[[(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]
 [(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]
 [(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]]
shift matrix
[(1, 6), (2, 5), (3, 4)]
final matrix
[[(7, 2) (1, 3) (2, 4) (3, 5) (4, 6) (5, 7) (6, 1)]
 [(6, 3) (7, 4) (1, 5) (2, 6) (3, 7) (4, 1) (5, 2)]
 [(5, 4) (6, 5) (7, 6) (1, 7) (2, 1) (3, 2) (4, 3)]]
```

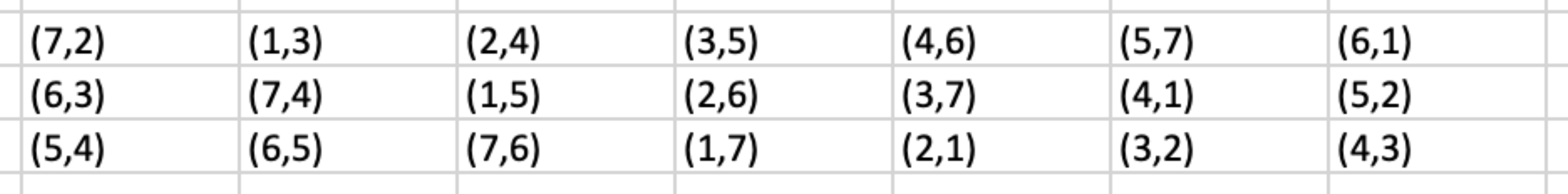


1 [1,2,3,4,5,6]

2 [1,2,3] [4,5,6]

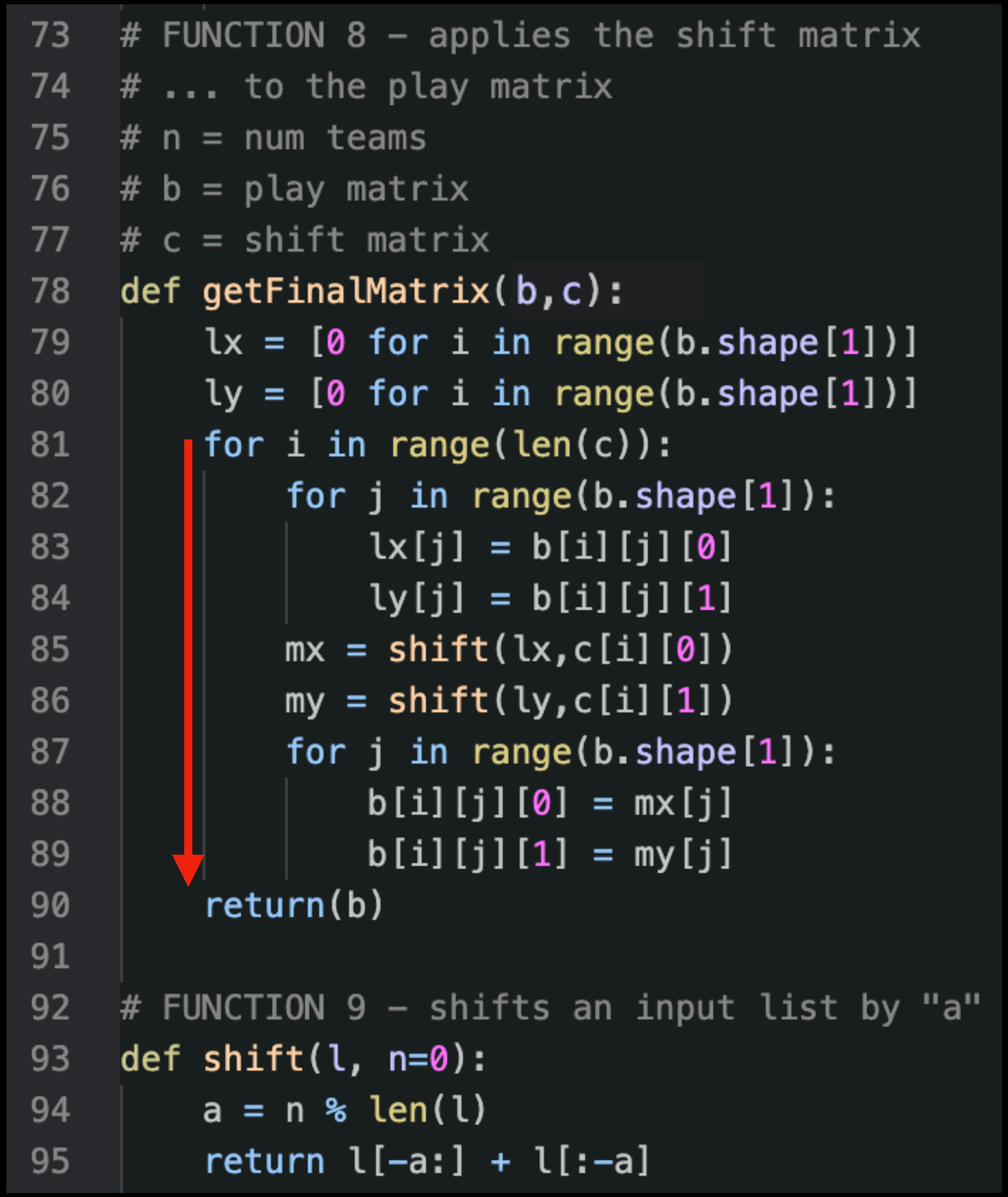
3 [1,2,3] [6,5,4]

4 [(1,6) (2,5) (3,4)]



(7,2)	(1,3)	(2,4)	(3,5)	(4,6)	(5,7)	(6,1)
(6,3)	(7,4)	(1,5)	(2,6)	(3,7)	(4,1)	(5,2)
(5,4)	(6,5)	(7,6)	(1,7)	(2,1)	(3,2)	(4,3)

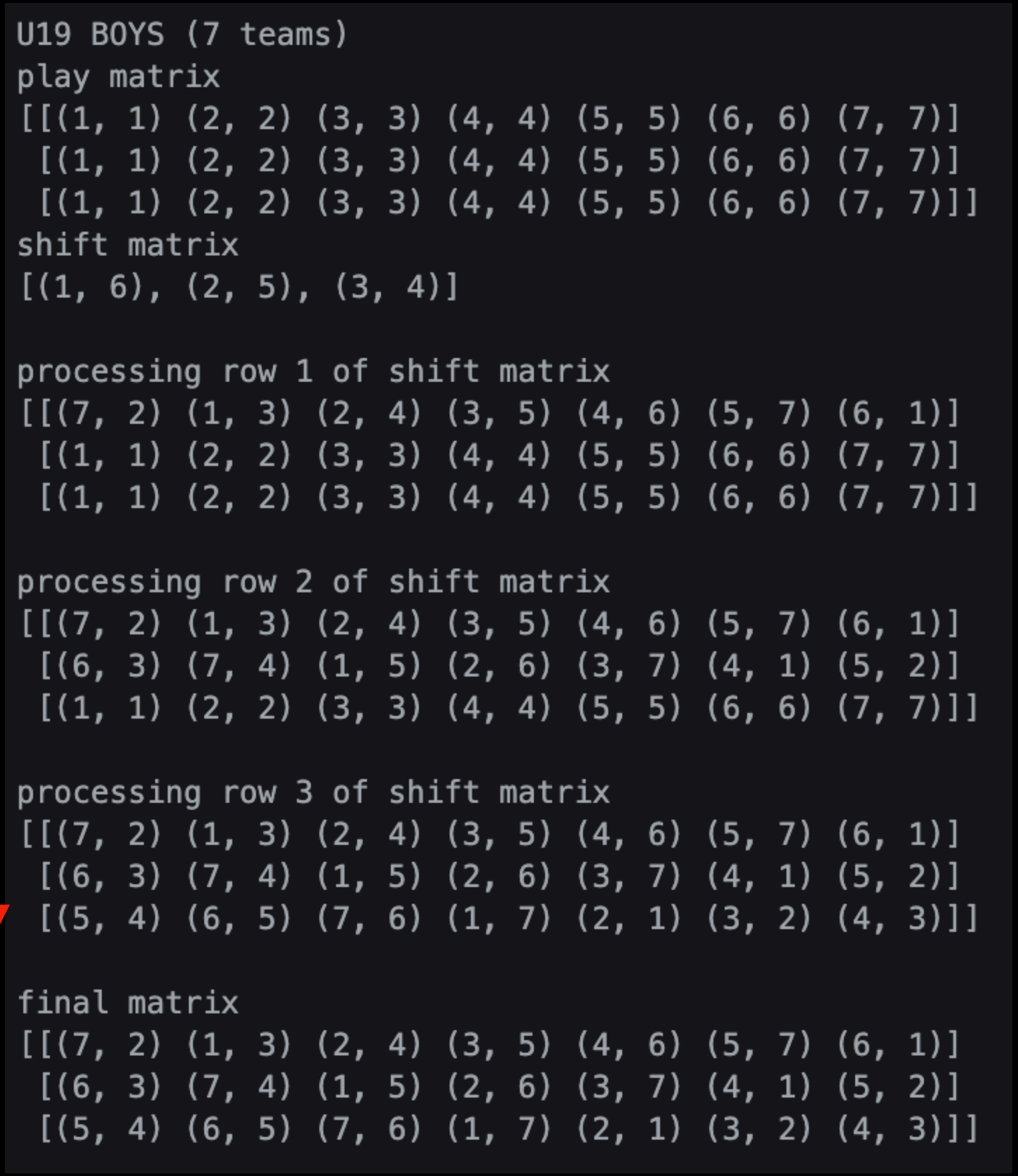
Function 8; “*getFinalMatrix*”



```python
# FUNCTION 8 - applies the shift matrix
# ... to the play matrix
# n = num teams
# b = play matrix
# c = shift matrix
def getFinalMatrix(b,c):
    lx = [0 for i in range(b.shape[1])]
    ly = [0 for i in range(b.shape[1])]
    for i in range(len(c)):
        for j in range(b.shape[1]):
            lx[j] = b[i][j][0]
            ly[j] = b[i][j][1]
        mx = shift(lx,c[i][0])
        my = shift(ly,c[i][1])
        for j in range(b.shape[1]):
            b[i][j][0] = mx[j]
            b[i][j][1] = my[j]
    return(b)

# FUNCTION 9 - shifts an input list by "a"
def shift(l, n=0):
    a = n % len(l)
    return l[-a:] + l[:-a]
```

As can be seen, the function iteratively applies the shift matrix to the play matrix.

Row-by-row the play matrix gets turned in to the final matrix.

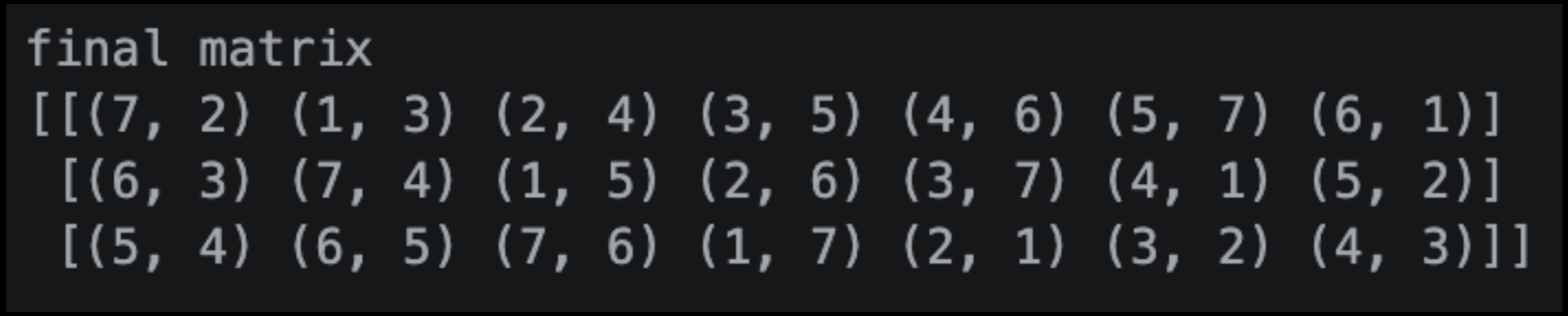


```
U19 BOYS (7 teams)
play matrix
[[(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]
 [(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]
 [(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]]
shift matrix
[(1, 6), (2, 5), (3, 4)]

processing row 1 of shift matrix
[[(7, 2) (1, 3) (2, 4) (3, 5) (4, 6) (5, 7) (6, 1)]
 [(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]
 [(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]]

processing row 2 of shift matrix
[[(7, 2) (1, 3) (2, 4) (3, 5) (4, 6) (5, 7) (6, 1)]
 [(6, 3) (7, 4) (1, 5) (2, 6) (3, 7) (4, 1) (5, 2)]
 [(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6) (7, 7)]]

processing row 3 of shift matrix
[[(7, 2) (1, 3) (2, 4) (3, 5) (4, 6) (5, 7) (6, 1)]
 [(6, 3) (7, 4) (1, 5) (2, 6) (3, 7) (4, 1) (5, 2)]
 [(5, 4) (6, 5) (7, 6) (1, 7) (2, 1) (3, 2) (4, 3)]]

final matrix
[[(7, 2) (1, 3) (2, 4) (3, 5) (4, 6) (5, 7) (6, 1)]
 [(6, 3) (7, 4) (1, 5) (2, 6) (3, 7) (4, 1) (5, 2)]
 [(5, 4) (6, 5) (7, 6) (1, 7) (2, 1) (3, 2) (4, 3)]]
```

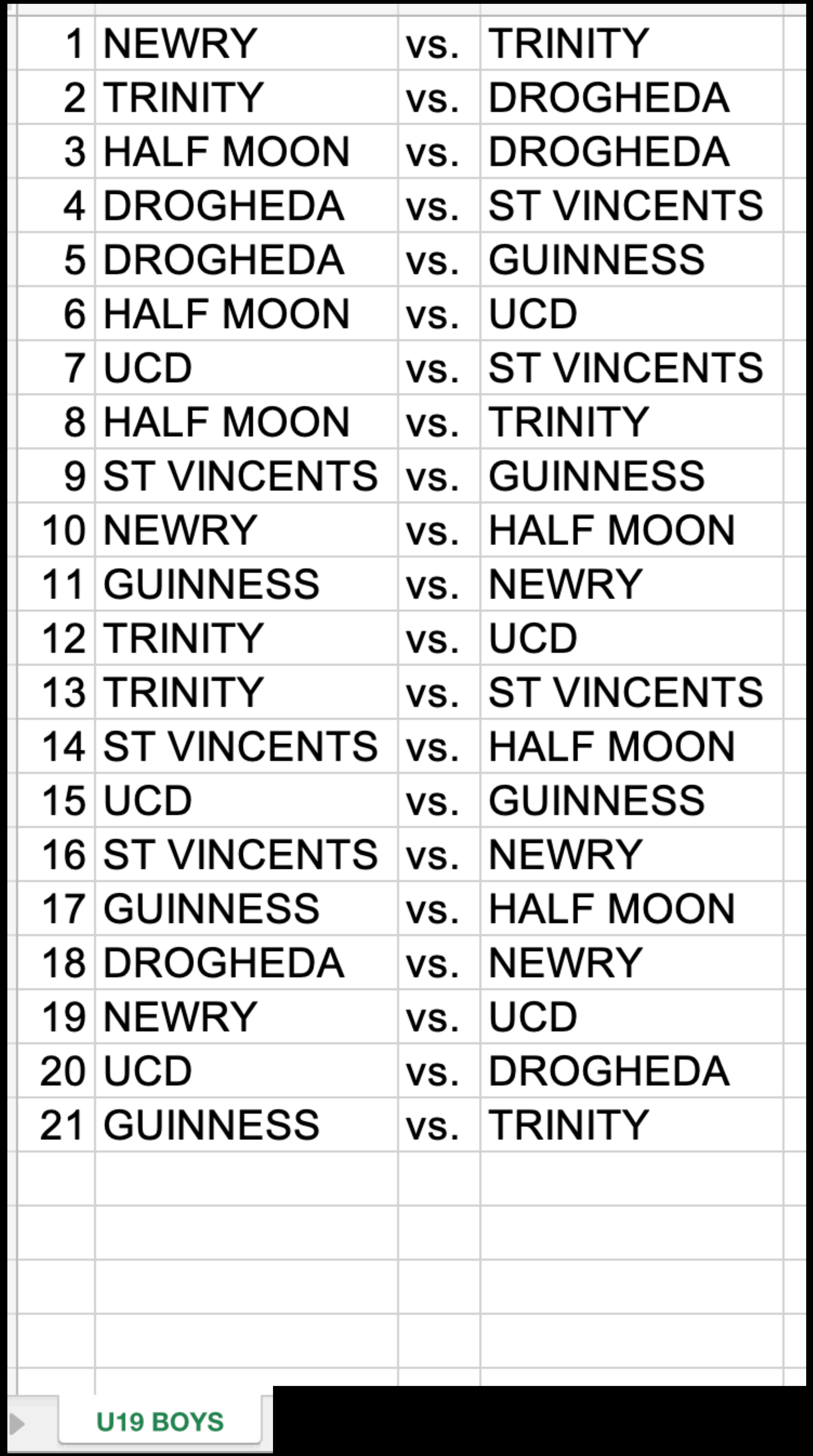
9. Conclusion

The final matrix is achieved; a list of coordinate pairs corresponding to which teams need to play one another to fulfil a complete round-robin roster of games for the water polo season.
To take the example we have used, the U19B (under 19 years old boys league), we see that each of the seven teams that entered that competition plays every other team once.

The code can be configured to make this a *double round robin* (i.e. each teams plays every other team but twice).



```
final matrix
[[(7, 2) (1, 3) (2, 4) (3, 5) (4, 6) (5, 7) (6, 1)]
 [(6, 3) (7, 4) (1, 5) (2, 6) (3, 7) (4, 1) (5, 2)]
 [(5, 4) (6, 5) (7, 6) (1, 7) (2, 1) (3, 2) (4, 3)]]
```



1	NEWRY	vs.	TRINITY
2	TRINITY	vs.	DROGHEDA
3	HALF MOON	vs.	DROGHEDA
4	DROGHEDA	vs.	ST VINCENTS
5	DROGHEDA	vs.	GUINNESS
6	HALF MOON	vs.	UCD
7	UCD	vs.	ST VINCENTS
8	HALF MOON	vs.	TRINITY
9	ST VINCENTS	vs.	GUINNESS
10	NEWRY	vs.	HALF MOON
11	GUINNESS	vs.	NEWRY
12	TRINITY	vs.	UCD
13	TRINITY	vs.	ST VINCENTS
14	ST VINCENTS	vs.	HALF MOON
15	UCD	vs.	GUINNESS
16	ST VINCENTS	vs.	NEWRY
17	GUINNESS	vs.	HALF MOON
18	DROGHEDA	vs.	NEWRY
19	NEWRY	vs.	UCD
20	UCD	vs.	DROGHEDA
21	GUINNESS	vs.	TRINITY

Thank You