Understanding the problem statement

Input matrix : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 Reverse 90 degrees in anticlockwise direction Output : 4 8 12 16 3 7 11 15 2 6 10 14 1 5 9 13

Algorithm

1. Reverse rows of matrix O(n^2) 4 3 2 1 8 7 6 5 12 11 10 9 16 15 14 13 2. Transpose the above matrix O(n^2) 4 8 12 16 3 7 11 15 2 6 10 14 1 5 9 13

Complexity

Time :

```
O(n^2)
```

## Space: O(1)

Implementation

In [18]:

```python
def swap(a,b):
    temp = a
    a = b
    b = temp
    return a,b

def reverserow(arr,size):
    rows = size
    cols = size
    for i in range(rows):
        k = cols-1
        for j in range(cols):
            if j>k:
                break
            arr[i][j],arr[i][k] = swap(arr[i][j],arr[i][k])
            k-=1
    return arr

def transpose(arr,size):
    rows = size
    cols = size
    for i in range(rows):
        for j in range(i,cols):
            arr[i][j],arr[j][i] = swap(arr[i][j],arr[j][i])
    return arr


def display(arr,size):
    rows = size
    cols = size
    for i in range(rows):
        print("\n")
        for j in range(cols):
            print(arr[i][j],end="\t")
    return arr
```

In [23]:

```
def rotateMatrix(arr,size):
    display(arr,size)
    print("\n")
    print("***************************ROTATED********************")
    arr = reverserow(arr,size)
    arr = transpose(arr,size)
    display(arr,size)
```

In [24]:

```
arr = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
size = 4
rotateMatrix(arr,size)
```

```
1       2       3       4

5       6       7       8

9       10      11      12

13      14      15      16

***************************ROTATED********************


4       8       12      16

3       7       11      15

2       6       10      14

1       5       9       13
```

In [ ]: