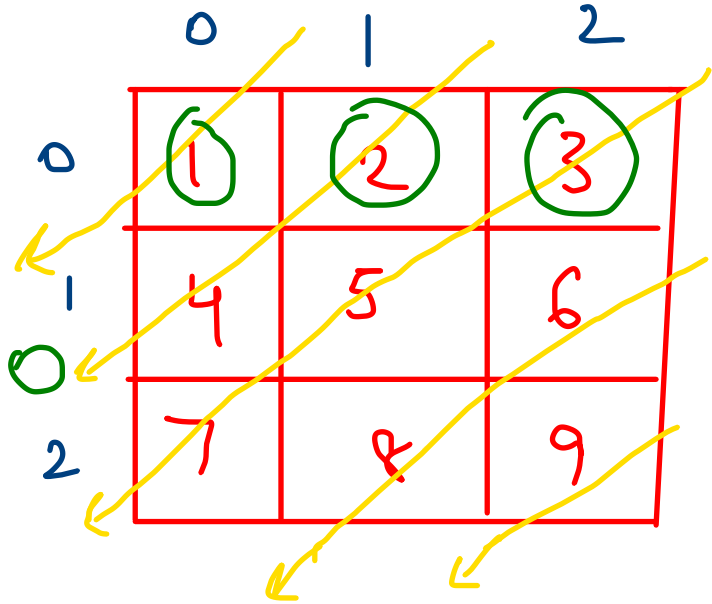
Print the matrix left-diagonal wise

n = 3




```java
public static void leftDia(int arr[][], int n){

    for(int gap = 0; gap < n; gap++){
        for(int i = 0, j = gap; j >= 0; i++, j--){
            System.out.print(arr[i][j] + " ");
        }
    }

    for(int gap = 1; gap < n; gap++){
        for(int i = gap, j = n-1; i < n; i++,j--){
            System.out.print(arr[i][j] + " ");
        }
    }
```

TC - O(n×n) SC - O(1)

ans = 1 2 4 3 5 7 6 8 9

(i, j)

gap = 0
(0 0) = 1
(1 -1) X

gap = 1
(0, 1) = 2
(1, 0) = 4
(2 -1) X

gap = 1
(1 2) = 6

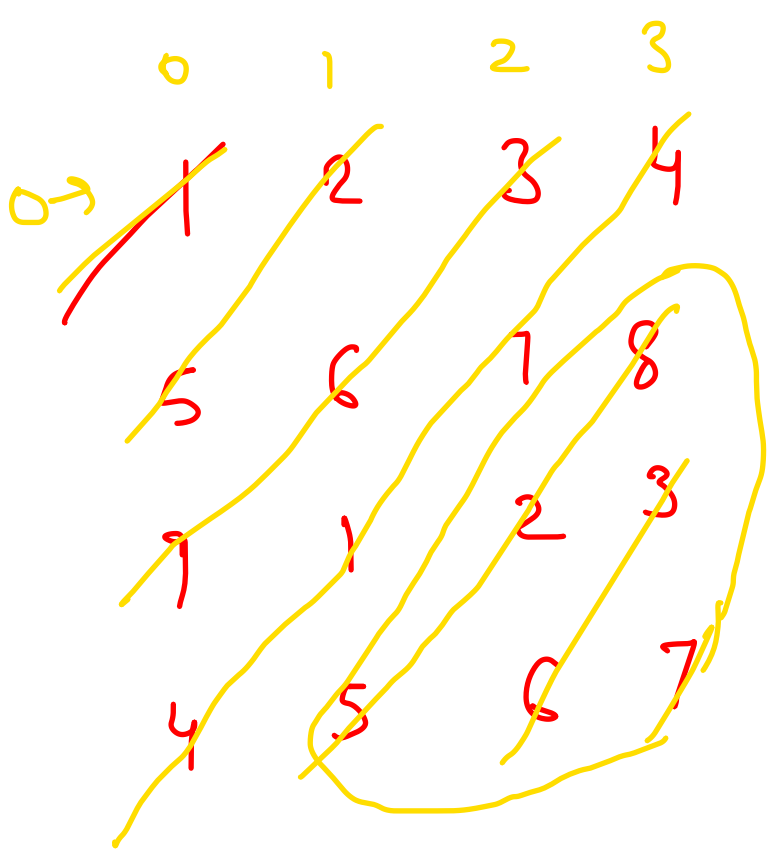
gap = 2
(0 2) = 3
(1 1) = 5
(2 0) = 7
(3, -1) X

gap = 2
(2 1) = 8
(3 0) X

gap = 3
(3 < 3) X

gap = 2
(2 2) = 9
(3 1) = X

gap = 3
(3 < 3) X



i=0

for(int =0; ; i++){

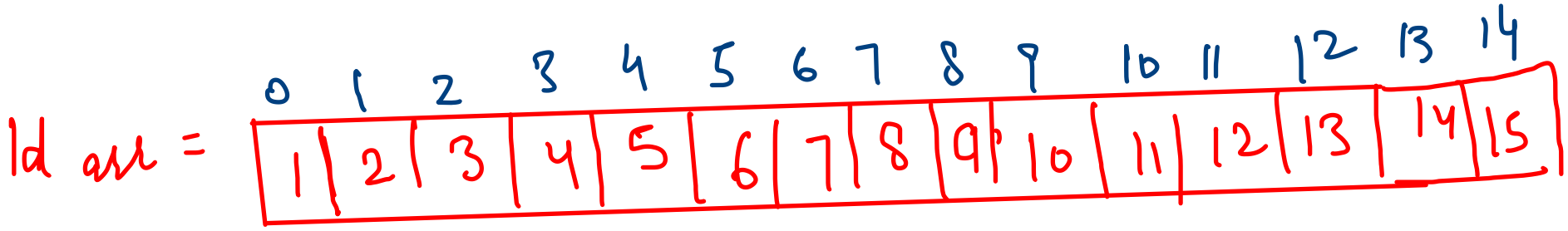
for(int j= gap ; j<n ; j--){

Approach 2

using while loop

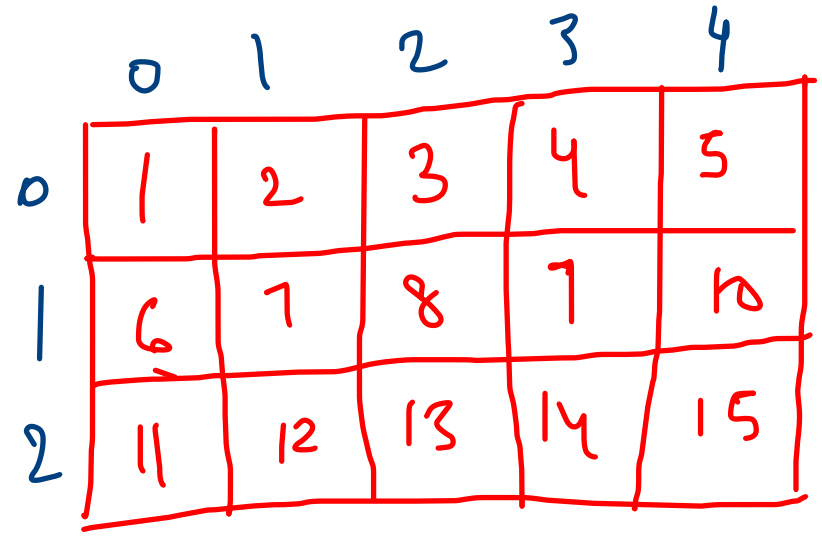
```
public static void leftDia(int arr[][], int n){

    for(int gap = 0; gap < n; gap++){
        //for(int i = 0, j = gap; j >= 0; i++, j--){

            int i = 0;
            int j =gap;
            while(j >=0)
            System.out.print(arr[i][j] + " ");
            i++;
            j--;
        }
    }

    for(int gap = 1; gap < n; gap++){
        for(int i = gap, j = n-1; i < n; i++,j--){
            System.out.print(arr[i][j] + " ");
        }
    }
```

Convert 1-D Array to 2-D Array



p = 3 // rows

q = 5 // cols.



(idx) 1d	2d (i,j)
0	(0,0)
1	(0,1)
2	(0,2)
3	(0,3)
4	(0,4)
5	(1,0)
6	(1,1)
7	(1,2)
8	(1,3)
9	(1,4)
10	(2,0)
11	(2,1)
12	(2,2)
13	(2,3)
14	(2,4)

1d → idx

2d → i & j

```
i = idx / q ;
j = idx % q ;
```

V.V. Imp

find indexo of 2d using 1d

```
i = idx/q;
j = idx %q;
```

find indexo of 1d using 2d

```
idx = i x q +j
```

```java
public static int[][] conver2D(int arr1d[], int n, int p, int q){
    int arr2d[][] = new int[p][q];

    for(int idx = 0; idx < n; idx++){
        int i = idx / q;
        int j = idx % q;

        arr2d[i][j] = arr1d[idx];
    }
    return arr2d;
}
```

TC - O(n)

or

O(p*q)

SC - O(p*q)

0 1 2 3

0 1 2 3 4 → $k=2$ → 3 4 1 2

1 5 6 7 8 → 7 8 5 6

2 9 10 11 12 → 11 12 9 10

3 13 14 15 16 → 15 16 13 14

$k=0$ 1 2 3 4

$k=1$ 4 1 2 3

$k=2$ 3 4 1 2

1)

```
    Scanner s  = new Scanner(System.in);
    int n = s.nextInt();

    int arr[][] = new int[n][n];
    for(int i = 0; i < n; i++){
        for(int j = 0; j <n;j++){
            arr[i][j] = s.nextInt();
        }
    }
    int k = s.nextInt();

    rowWise(arr,n,k);

}
```

3)

```
    public static void reverse(int arr[], int i, int j){
        while(i < j){
            swap(arr, i, j);
            i++;
            j--;
        }
    }
    public static void swap(int arr[], int x, int y){
        int temp = arr[x];
        arr[x] = arr[y];
        arr[y] = temp;
    }
}
```

n

1

2)

```
public static void rowWise(int arr[][], int n, int k){
    k = -1 * k; // to submit ques

    for(int i = 0; i < n; i++){
        k = k + n;
        k = k % n;

        reverse(arr[i], n - k, n -1);
        reverse(arr[i], 0, n - k -1);
        reverse(arr[i], 0, n-1);
    }
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            System.out.print(arr[i][j] +  " ");
        }
        System.out.println();
    }
}
```

n

n^2

TC → $O(n \times n)$

SC → $O(1)$

+2 = = -2

k = k + n

k = -2 + 4

k = 2