Convert 1-D Array to 2-D Array

arr1d =

1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11

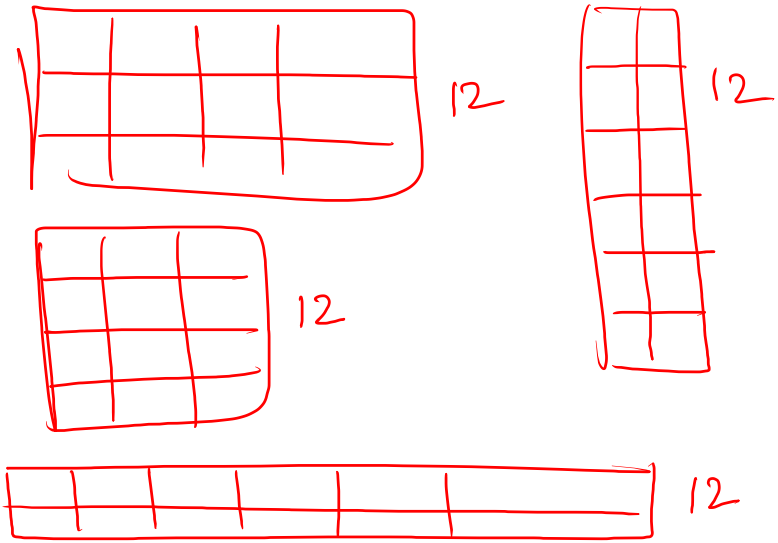
n = 12

p = 3 // no. of rows

q = 4 // no. of cols

↳ possibilities

- ↳ 3 * 4
- ↳ 4 * 3
- ↳ 2 * 6
- ↳ 6 * 2



arr1d =

1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11

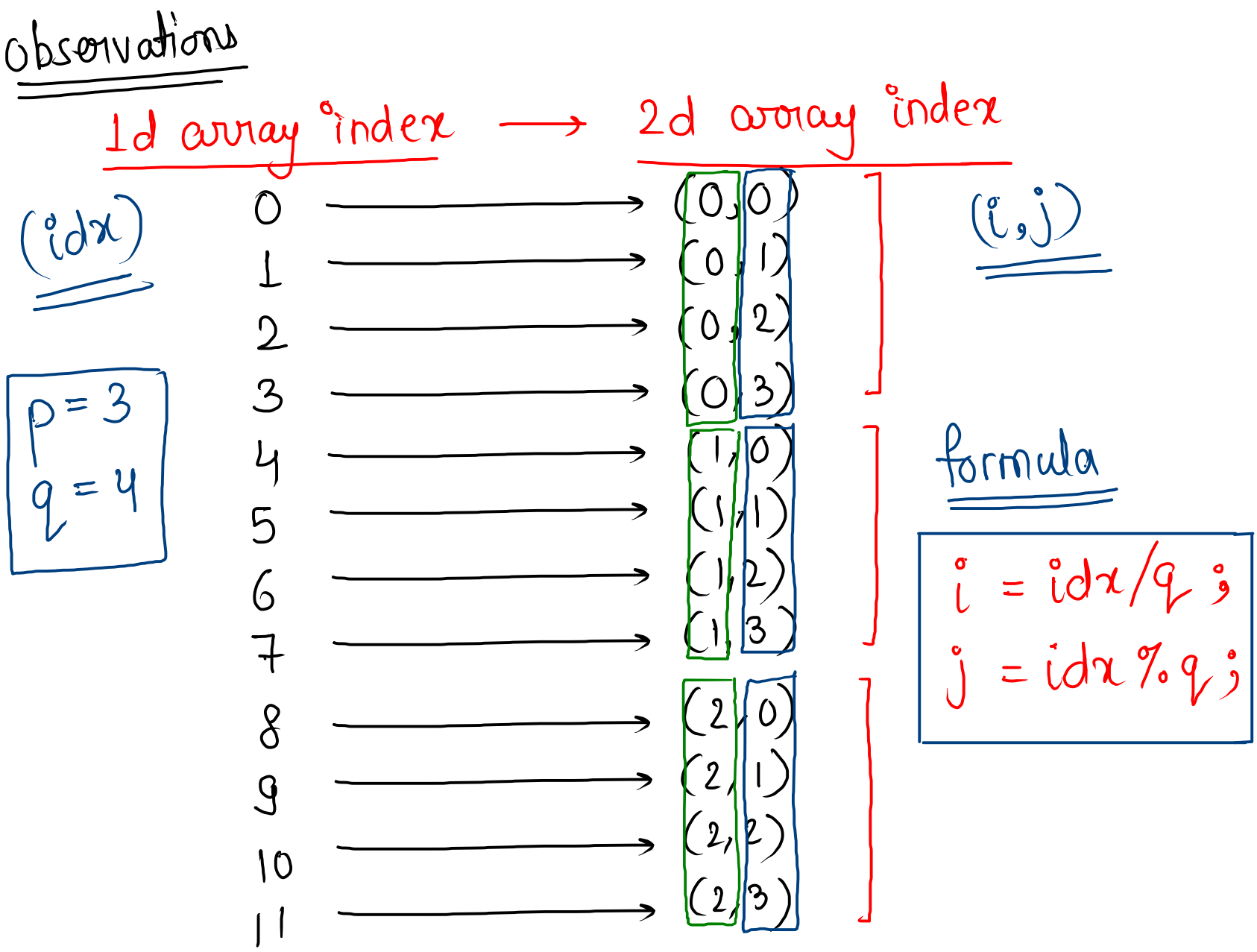
$p = 3$

$q = 4$

n

arr2d =

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12


Observations
1d array index → 2d array index
(idx)
(i,j)
0 → (0,0)
1 → (0,1)
2 → (0,2)
3 → (0,3)
4 → (1,0)
5 → (1,1)
6 → (1,2)
7 → (1,3)
8 → (2,0)
9 → (2,1)
10 → (2,2)
11 → (2,3)
p = 3
q = 4
formula
i = idx/q;
j = idx%q;

idx = 0 , i = 0/4 = 0
j = 0%4 = 0

idx = 1 , i = 1/4 = 0
j = 1%4 = 1

idx = 2 , i = 2/4 = 0
j = 2%4 = 2

idx = 3 , i = 3/4 = 0
j = 3%4 = 3

idx = 4 , i = 4/4 = 1
j = 4%4 = 0

idx = 5 , i = 5/4 = 1
j = 5%4 = 1

idx = 6 , i = 6/4 = 1
j = 6%4 = 2

idx = 7 , i = 7/4 = 1
j = 7%4 = 3

idx = 8 , i = 8/4 = 2
j = 8%4 = 0

idx = 9 , i = 9/4 = 2
j = 9%4 = 1

idx = 10 , i = 10/4 = 2
j = 10%4 = 2

idx = 11 , i = 11/4 = 2
j = 11%4 = 3

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr1d = new int[n];
    for (int i = 0; i < n; i++) {
        arr1d[i] = scn.nextInt();
    }
    int p = scn.nextInt();
    int q = scn.nextInt();

    int[][] arr2d = convert1Dto2D(arr1d, p, q, n);
    for (int i = 0; i < p; i++) {
        for (int j = 0; j < q; j++) {
            System.out.print(arr2d[i][j] + " ");
        }
        System.out.println();
    }
}
public static int[][] convert1Dto2D(int[] arr1d, int p, int q, int n) {
    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;
}
```

T.C = O(n)

n = size of 1d array

T.C = O(p*q)

linear

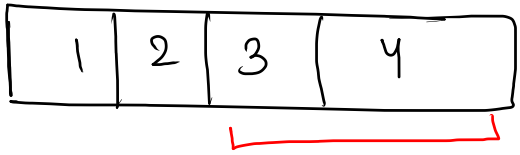
Shift Matrix Row-Wise (rotate each row by K)

n = 4

arr =

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

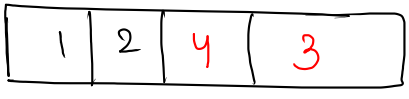
K = 2



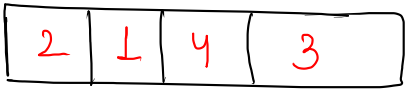
K = 2

trick

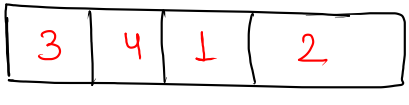
step1 :- reverse K elements from last



step2 :- reverse remaining elements



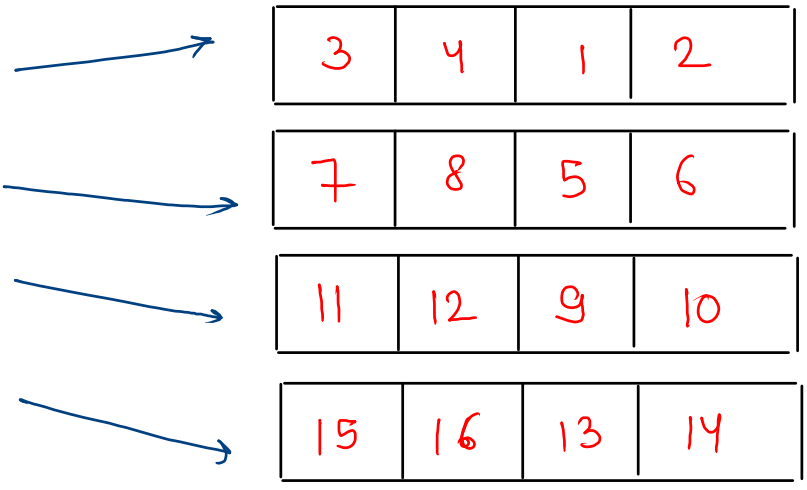
step3 :- reverse entire array



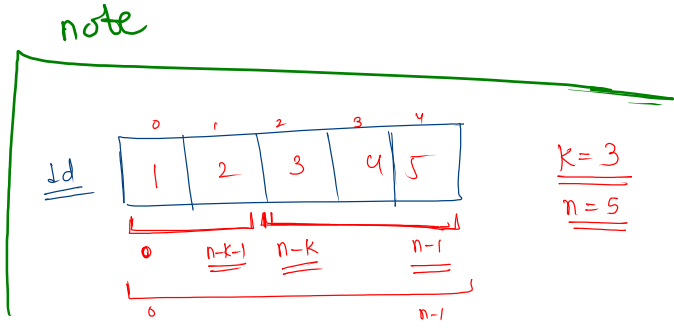
n = 4

arr =

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16



note



arr

1	2	3	4	5

$n = 5$

$k = -2$

$k = k + n$

$= (-2) + (5)$

$k = 3$

$k = 1$ (5 1 2 3 4)

$k = 2$ (4 5 1 2 3)

$k = 3$ (3 4 5 1 2)

$k = 4$ (2 3 4 5 1)

$k = 5$ (1 2 3 4 5)

$k = -1$ (2 3 4 5 1)

$k = -2$ (3 4 5 1 2)

$k = -3$ (4 5 1 2 3)

$k = -4$ (5 1 2 3 4)

$k = -5$ (1 2 3 4 5)

Code

```java
public static void shiftMatrixByK(int[][] arr, int n, int k) {
    k = -1 * k; // just to submit question
    for (int i = 0; i < n; i++) {
        k = k + n; // handle -ve k values
        k = k % n; // handle rotation
        reverse(arr[i], n - k, n - 1);
        reverse(arr[i], 0, n - k - 1);
        reverse(arr[i], 0, n - 1);
    }
}

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

→ not required in actual solution

$T.C = O(n^2)$