(i,j)

	0	1	2
0	(0,0)	(0,1)	(0,2)
1	(1,0)	(1,1)	(1,2)
2	(2,0)	(2,1)	(2,2)

i = index/col_size;

j = index%col_size;

idx		i		j
0	→	0/3 = 0		0%3 = 0
1	→	1/3 = 0	H	1%3 = 1
2	→	2/3 = 0		2%3 = 2
3	→	3/3 = 1		3%3 = 0
4	→	4/3 = 1	H	4%3 = 1
5	→	5/3 = 1		5%3 = 2
6	→	6/3 = 2		6%3 = 0
7	→	7/3 = 2	H	7%3 = 1
8	→	8/3 = 2		8%3 = 2

same

same

0	1	2	3	4	5	6	7	8

Shift Matrix Row-Wise

arr

	0	1	2	3
0	1	2	3	4
1	3	2	4	1
2	5	7	6	2
3	3	9	8	6

$n = 4$

$k = -7$

0	1	2	3
1	2	3	4

$$\begin{cases} k = k \% n \\ k = -7 \% 4 \\ k = 1 \end{cases}$$

✓1) $\boxed{k = k + n}$

$= -7 + 4 = -3$

✓2) $\boxed{k = k \% n}$

$= -3 \% 4$

$= 1$

arr

	0	1	2	3
0	4	1	2	3
1	1	3	2	4
2	2	5	7	6
3	6	3	9	8

modified

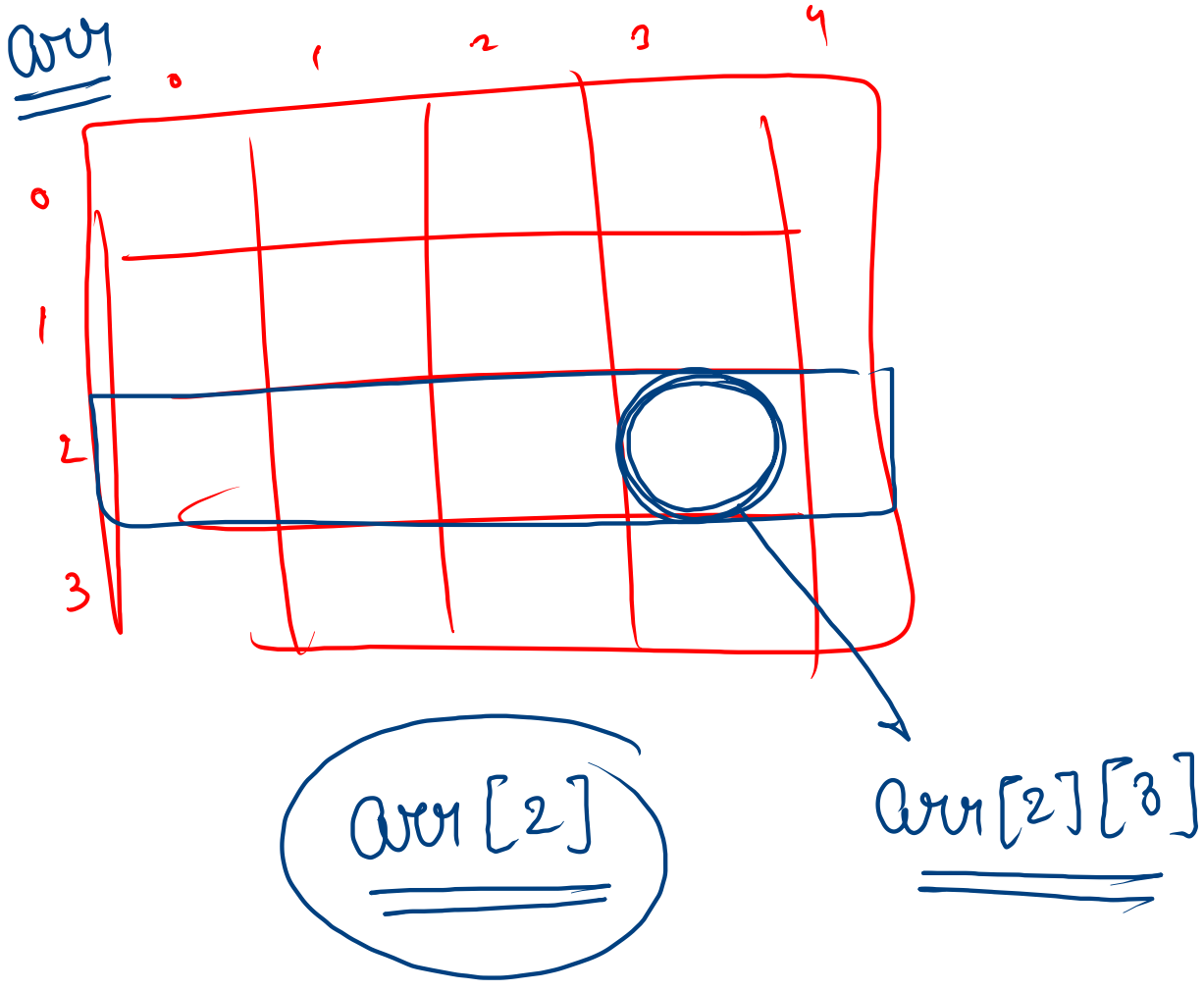
1 2 3 4	$k = 0$
$\boxed{4\ 1\ 2\ 3}$	$k = -1$
3 4 1 2	$k = -2$
$\boxed{2\ 3\ 4\ 1}$	$k = -3$ ✓
1 2 3 4	$k = -4$
4 1 2 3	$k = -5$
3 4 1 2	$k = -6$
$\boxed{2\ 3\ 4\ 1}$	$k = -7$ ✓

2	3	4	1
3	4	1	2
4	1	2	3
1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

Code

```java
public static int[][] shiftMatrix(int[][] arr, int n, int k) {
    k = k * -1;
    for (int row = 0; row < n; row++) {
        k = k + n;  // convert anti clockwise direction to clock wise
        k = k % n;  // convert larger k value to smaller k value

        // reverse last k elements
        reverse( arr[row], n - k, n - 1 );
        // reverse remainng elements
        reverse( arr[row], 0, n - k - 1 );
        // reverse entire 1d array
        reverse( arr[row], 0, n - 1 );
    }
    return arr;
}
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 i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```


arr
0
1
2
3
4
0
1
2
3
arr[2]
arr[2][3]

Modify The Matrix

	0	1	0	1	0	0	1	0

(col)

		0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0	0
1	3	0	0	0	0	0	0	1	0
0	4	0	0	0	0	0	0	0	0
0	5	0	0	0	0	0	0	0	0
0	6	0	0	0	0	0	0	0	0
1	7	0	0	0	1	0	0	0	0

(row)

indexes available → (1,1), (3,6), (7,3)

ans

	0	1	2	3	4	5	6	7
0		1		1			1	
1	1	1	1	1	1	1	1	1
2		1		1			1	
3	1	1	1	1	1	1	1	1
4		1		1			1	
5		1		1			1	
6		1		1			1	
7	1	1	1	1	1	1	1	1

psudo code

i/p 2D array (m x n)

1) create 1D array row of size m
2) create 1D array col of size n
3) traverse in 2D array
   3.1) if arr[i][j] is 1
        then row[i] = 1 and col[j] = 1

4) traverse again in 2D array
   4.1) if row[i] == 1 or col[j] == 1
        then arr[i][j] = 1

Code

$$\boxed{\begin{aligned} T.C &= O(m*n) \\ S.C &= O(m+n) \end{aligned}}$$

```java
public static int[][] modifyMatrix(int[][] arr, int m, int n) {
    int[] row = new int[m];
    int[] col = new int[n];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if ( arr[i][j] == 1 ) {
                row[i] = 1;
                col[j] = 1;
            }
        }
    }
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if ( row[i] == 1 || col[j] == 1 ) {
                arr[i][j] = 1;
            }
        }
    }
    return arr;
}
```

⇒ Strings

str = "abcdef"

0 1 2 3 4 5

inbuilt

1) str.length(); // 6
2) str.charAt(5); // 'f'
3) str.toUpperCase();
4) str.toLowerCase();
5) concatenation

str = "abcdef";
0 1 2 3 4 5

syntex 1

str.substring(start_index, ending_index + 1);

ex:- str.substring(0, 3); // "abc"

str.substring(1, 4); // "bcd"

str.substring(0, 5); // "abcde"

str.substring(0, 6); // "abcdef"

str.substring(0, 7); // string index out of bound exception

str = "abc123";
0 1 2 3 4 5

syntex 2

str.substring(start_index);

ex:-

str.substring(3); // "123"

str.substring(5); // "3"

str.substring(6); // error

str.substring(0); // "abc123"