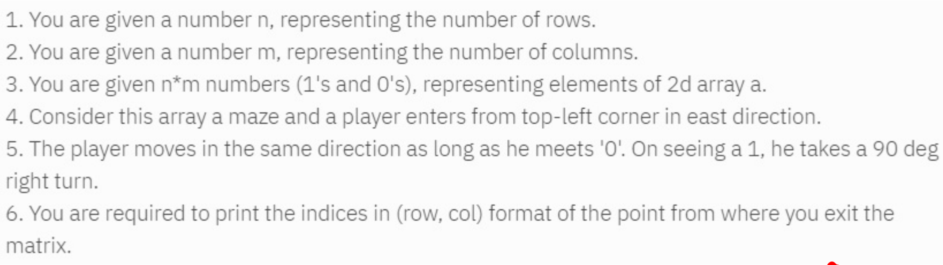
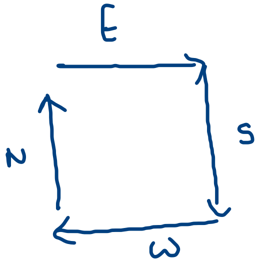
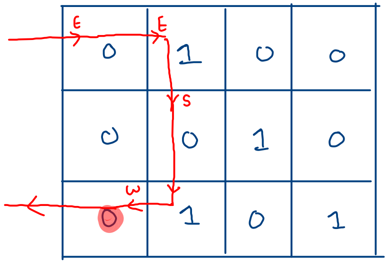
exit point

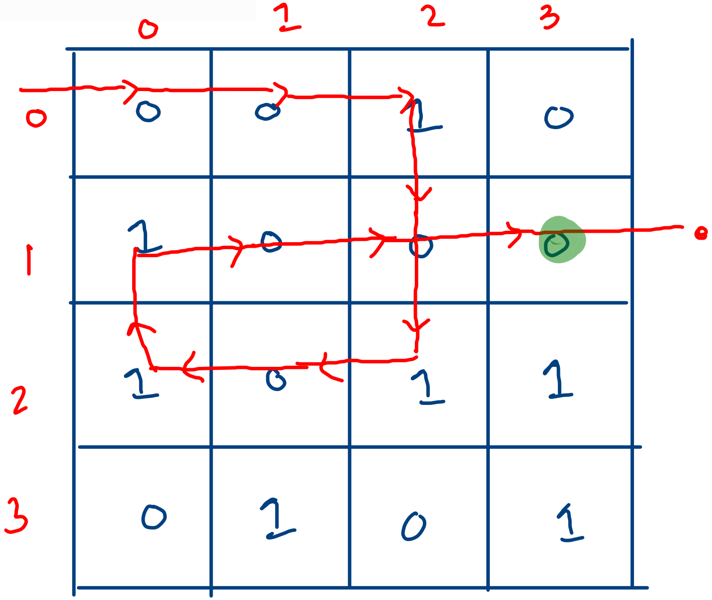


1. You are given a number n, representing the number of rows.
2. You are given a number m, representing the number of columns.
3. You are given n*m numbers (1's and 0's), representing elements of 2d array a.
4. Consider this array a maze and a player enters from top-left corner in east direction.
5. The player moves in the same direction as long as he meets '0'. On seeing a 1, he takes a 90 deg right turn.
6. You are required to print the indices in (row, col) format of the point from where you exit the matrix.

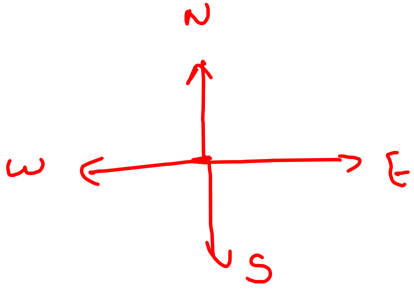


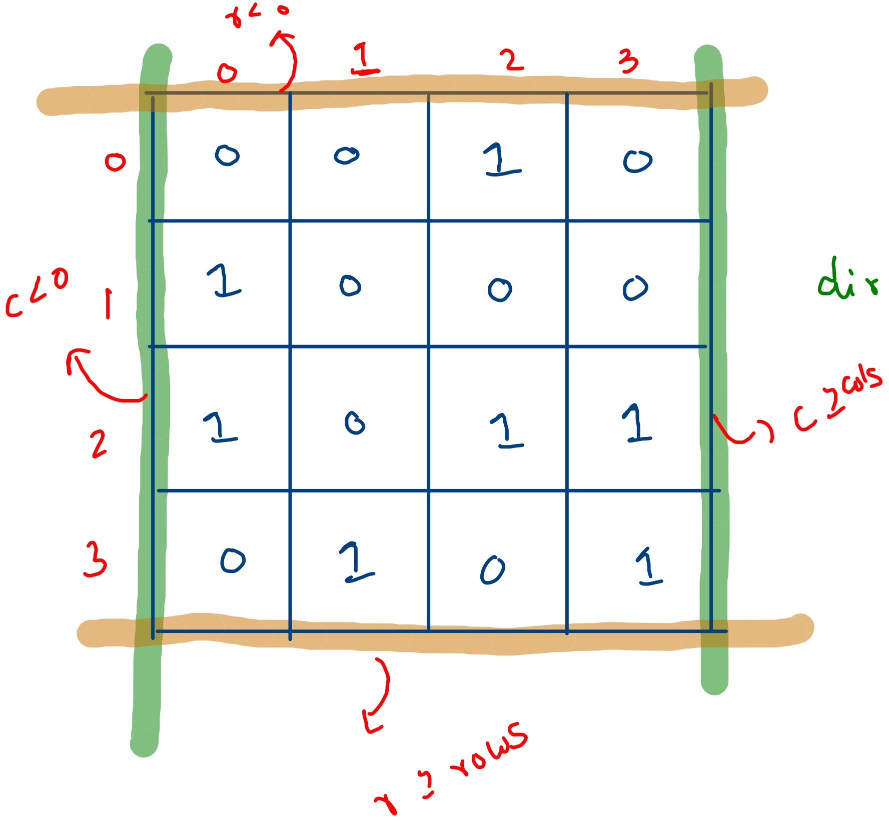


0	1	0	0
0	0	1	0
0	1	0	1



	0	1	2	3
0	0	0	1	0
1	1	0	0	0
2	1	0	1	1
3	0	1	0	1





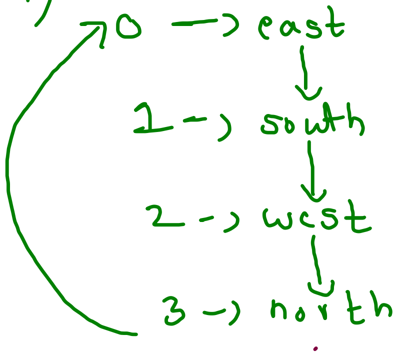
	0	1	2	3
0	0	0	1	0
1	1	0	0	0
2	1	0	1	1
3	0	1	0	1

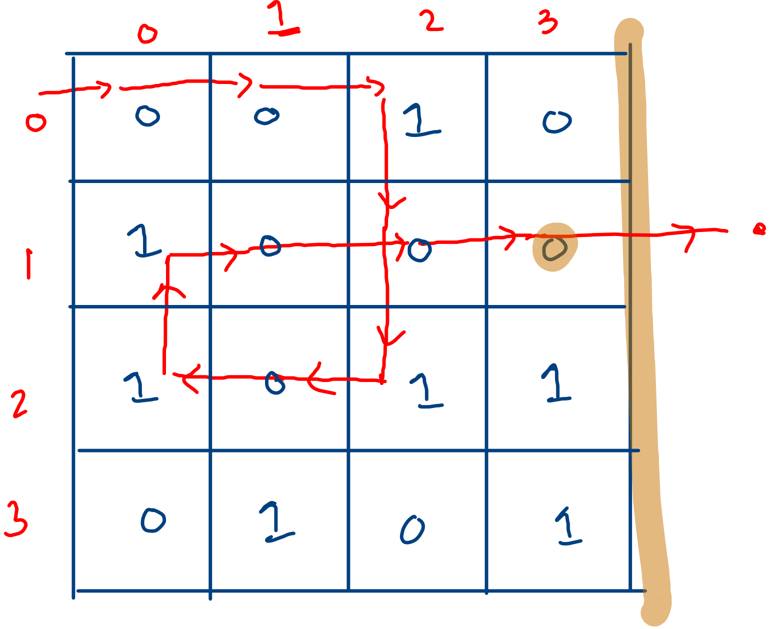
r = 0   dir = 0

c = 0

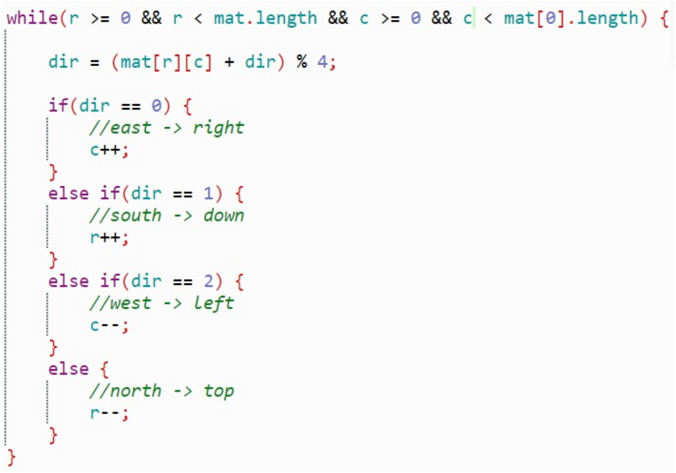
```
dir = (arr[r][c] + dir) % 4;
```

```
dir -> 0  (east) {
    c++;
}
dir -> 1  (south) {
    r++;
}
dir -> 2  (west) {
    c--;
}
dir -> 3  (north) {
    r--;
}
```





	0	1	2	3
0	0	0	1	0
1	1	0	0	0
2	1	0	1	1
3	0	1	0	1



```
while(r >= 0 && r < mat.length && c >= 0 && c < mat[0].length) {

    dir = (mat[r][c] + dir) % 4;

    if(dir == 0) {
        //east -> right
        c++;
    }
    else if(dir == 1) {
        //south -> down
        r++;
    }
    else if(dir == 2) {
        //west -> left
        c--;
    }
    else {
        //north -> top
        r--;
    }
}
```

r = ~~0~~ ~~1~~ ~~2~~ 1, c = ~~0~~ ~~1~~ ~~2~~ ~~1~~ ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

dir = 0

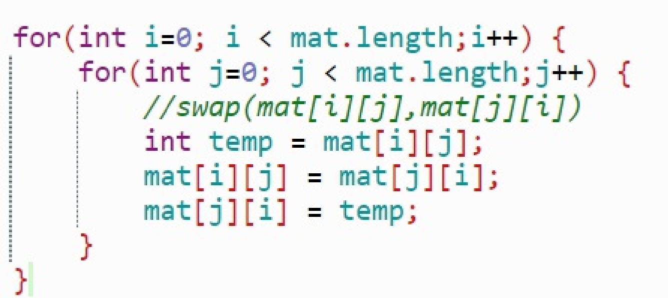
$$\begin{tikzcd}
\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & 8 & 9 \\ \hline \end{array} \arrow[rr, "\text{Clockwise } 90^\circ", "\text{rotation}"'] \arrow[dr, "\text{transpose}"'] & & \begin{array}{ccc} 0 & 1 & 2 \\ \hline 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{array} \\
& \begin{array}{ccc} 0 & 1 & 2 \\ \hline 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{array} \arrow[ur, "\text{column reversal}"] &
\end{tikzcd}$$

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

→

1	4	7
2	5	8
3	6	9

swap (mat [i] [j], mat [j] [i])

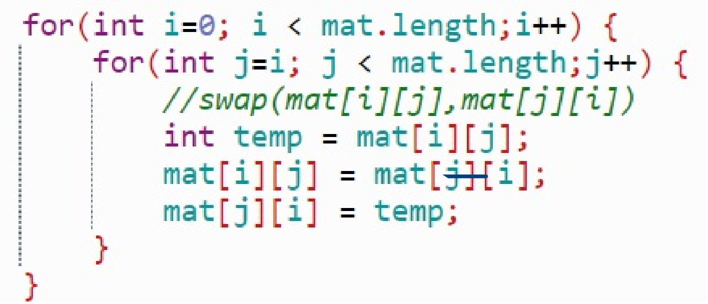


```
for(int i=0; i < mat.length;i++) {
    for(int j=0; j < mat.length;j++) {
        //swap(mat[i][j],mat[j][i])
        int temp = mat[i][j];
        mat[i][j] = mat[j][i];
        mat[j][i] = temp;
    }
}
```

i = 0 | j = 0, 1, 2 (0,0) (0,1), (0,2

i = 1 | j = 1, 2 (1,1) (1,2)

i = 2 | j = 2 (2,2)



```
for(int i=0; i < mat.length;i++) {
    for(int j=i; j < mat.length;j++) {
        //swap(mat[i][j],mat[j][i])
        int temp = mat[i][j];
        mat[i][j] = mat[~~j]~~[i];
        mat[j][i] = temp;
    }
}
```

i				
0	✓ (0,0)	✓ (0,1)	✓ (0,2)	✓ (0,3)
1	✗ (1,0)	✓ (1,1)	✓ (1,2)	✓ (1,3)
2	✗ (2,0)	✗ (2,1)	✓ (2,2)	(2,3)
3	✗ (3,0)	✗ (3,1)	✗ (3,2)	(3,3)

column reversal

	~~↓ lo~~	~~lo~~ hi	~~hi~~ lo	~~↓~~ hi
	0	1	2	3
0	m ~~a~~	j ~~e~~	~~i~~ e	~~m~~ a
1	n ~~b~~	~~f~~ j	~~j~~ f	~~n~~ b
2	o ~~c~~	~~g~~ k	~~k~~ g	~~o~~ c
3	p ~~d~~	~~h~~ l	~~l~~	~~p~~ d

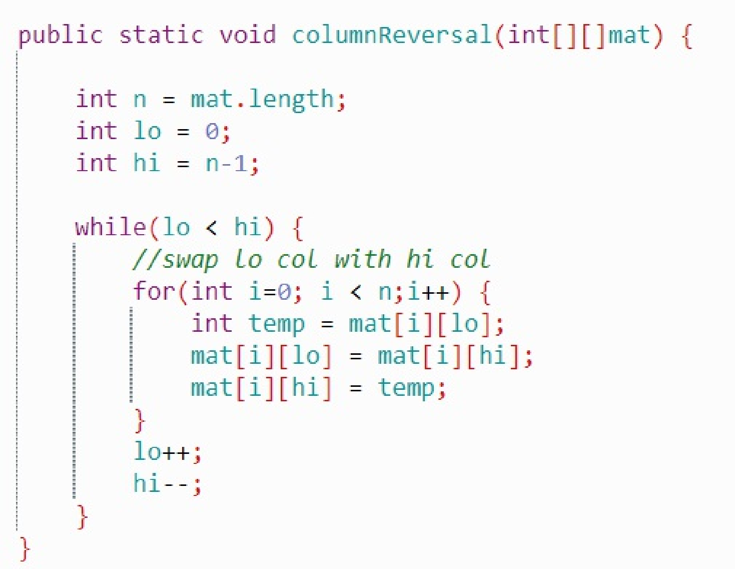
a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

tranpose

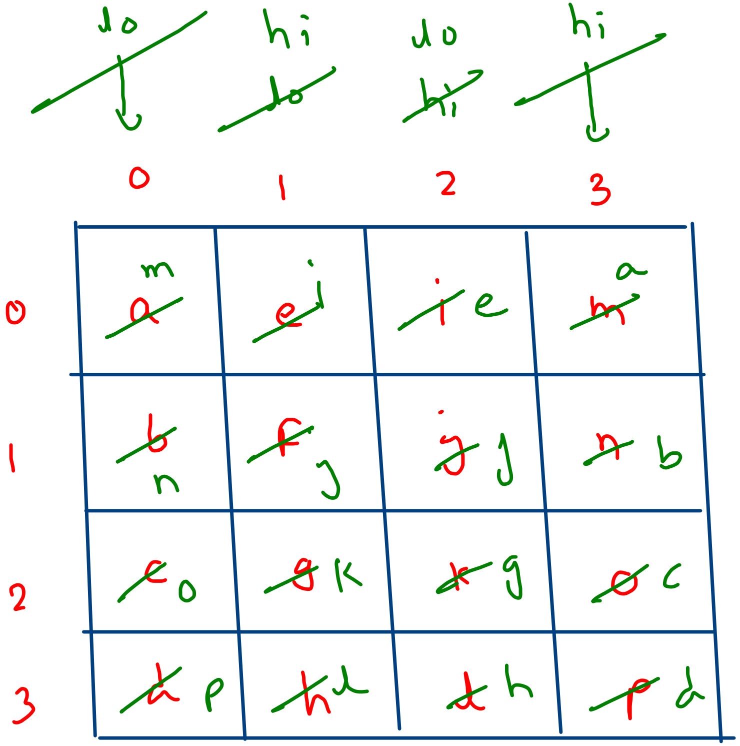
a	e	i	m
b	f	j	n
c	g	k	o
d	h	l	p

col reversal

m	i	e	a
n	j	f	b
o	k	g	c
p	l	h	d

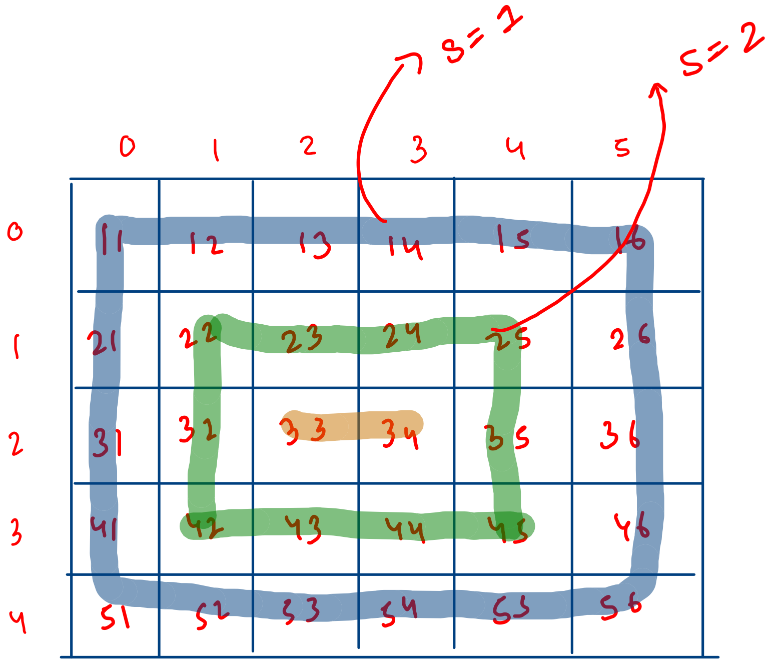


```
public static void columnReversal(int[][]mat) {

    int n = mat.length;
    int lo = 0;
    int hi = n-1;

    while(lo < hi) {
        //swap lo col with hi col
        for(int i=0; i < n;i++) {
            int temp = mat[i][lo];
            mat[i][lo] = mat[i][hi];
            mat[i][hi] = temp;
        }
        lo++;
        hi--;
    }
}
```



	0	1	2	3
0	~~a~~ m	~~e~~ i	~~i~~ e	~~m~~ a
1	~~b~~ n	~~f~~ j	~~j~~ j	~~n~~ b
2	~~c~~ o	~~g~~ k	~~k~~ g	~~o~~ c
3	~~d~~ p	~~h~~ d	~~d~~ h	~~p~~ d

Ring rotate



	0	1	2	3	4	5
0	11	12	13	14	15	16
1	21	22	23	24	25	26
2	31	32	33	34	35	36
3	41	42	43	44	45	46
4	51	52	53	54	55	56

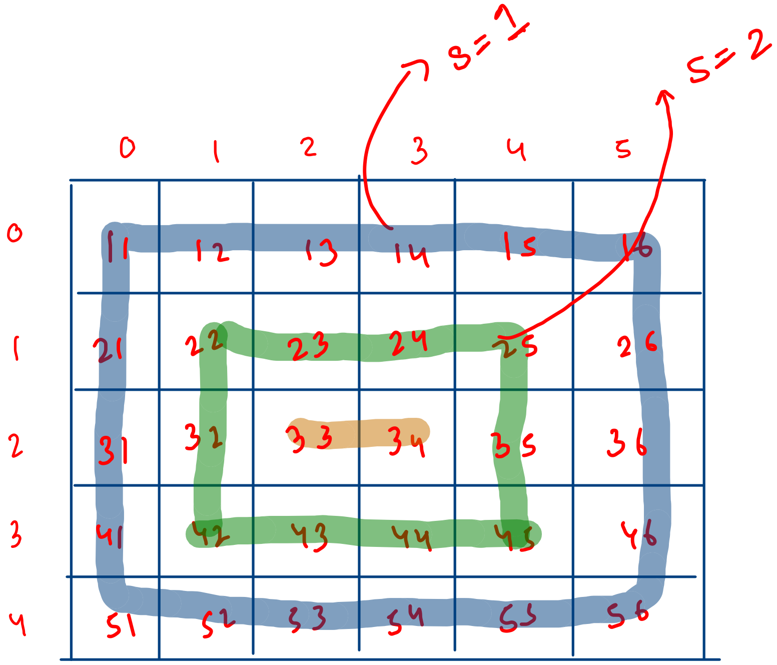
s=2, r= 3

	0	1	2	3	4	5
0	11	12	13	14	15	16
1	21					26
2	31		33	34		36
3	41					46
4	51	52	53	54	55	56

1. fill 1d array with s= 2 of matrix

2. rotate array r

3. fill s= 2 with 1d array



	0	1	2	3	4	5
0	11	12	13	14	15	16
1	21	22	23	24	25	26
2	31	32	33	34	35	36
3	41	42	43	44	45	46
4	51	52	53	54	55	56

1. fill 1d array with s= 2 of matrix
2. rotate array r
3. fill S= 2 with 1d array

s=2, r= 3

25	35	45	44
24	—	—	43
23	22	32	42

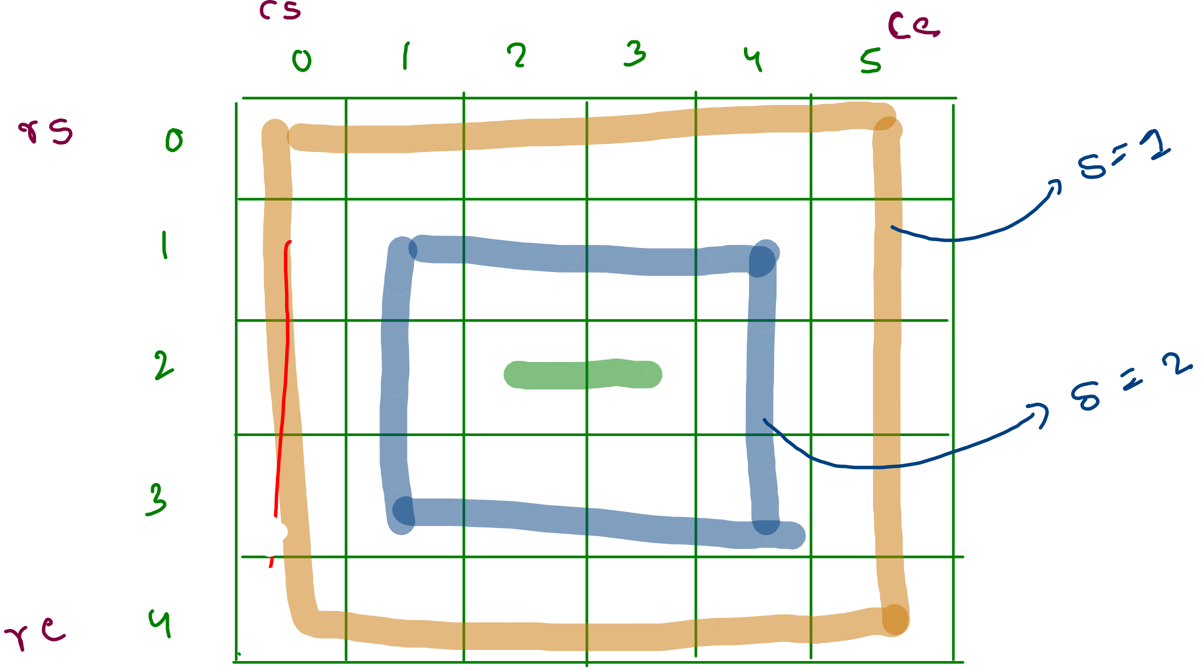
1.

22	32	42	43	44	45	35	25	24	23

2.

25	24	23	22	32	42	43	44	45	35

3.



rows = 5

cols = 6

rs = 0 + s - 1

cs = 0 + s - 1

re = rows - s

ce = cols - s

$$\boxed{t.e = 2*(re - rs) + 2*(ce - cs)}$$