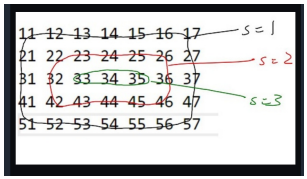
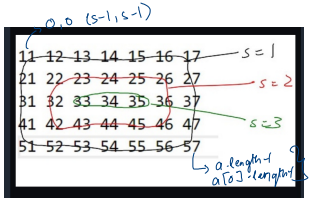
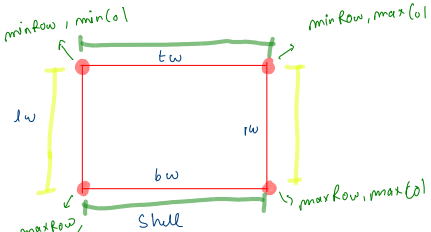
Ques) Shell Rotate



↳ fill the shell into a 1-D array
↳ Rotate the 1-D Array
↳ fill the array back to the shell



if shell no is s
Top left Corner ; s-1, s-1
Bottom Right Corner ; n-s, m-s



lw = rw
tw = bw

lw = maxRow - minRow + 1
bw = maxCol - minCol + 1

No of ele = lw + rw + tw + bw - 4
= 2*lw + 2*bw - 4
= 2[maxRow - minRow + 1]
+ 2[maxCol - minCol + 1] - 4

No of ele = 2maxRow - 2minRow + 2maxCol - 2minCol + 4
No of ele = 2x[maxRow - minRow + maxCol - minCol]

```java
public static void rotateShell(int[][] arr,int s, int k) {
    int[] a = fillOneDFromShell(arr,s);
    rotate(a,k);
    fillShellFromOneD(arr,s,a);
}
```

```java
public static int[] fillOneDFromShell(int[][] arr, int s) {
    int minr = s-1;
    int maxr = arr.length - s;
    int minc = s-1;
    int maxc = arr[0].length - s;

    int n = 2*(maxr-minr+maxc-minc);

    int[] oned = new int[n];

    //fill the array
    int count = 0;

    while(true) {

        //Left wall
        for(int i=minr;i<=maxr;i++) {
            oned[count] = arr[i][minc];
            count++;
            if(count == n) {
                return oned;
            }
        }
        minc++;

        //bottom wall
        for(int j=minc;j<=maxc;j++) {
            oned[count] = arr[maxr][j];
            count++;
            if(count == n) {
                return oned;
            }
        }
        maxr--;

        //right wall
        for(int i=maxr;i>=minr;i--) {
            oned[count] = arr[i][maxc];
            count++;
            if(count == n) {
                return oned;
            }
        }
        maxc--;

        //top wall
        for(int j=maxc;j>=minc;j--) {
            oned[count] = arr[minr][j];
            count++;
            if(count == n) {
                return oned;
            }
        }
        minr++;
    }
}
```

```java
public static void fillShellFromOneD(int[][] arr,int s, int[] oned) {
    int minr = s-1;
    int maxr = arr.length - s;
    int minc = s-1;
    int maxc = arr[0].length - s;

    int count = 0;
    int n = 2*(maxr-minr+maxc-minc);

    while(true) {

        //Left wall
        for(int i=minr;i<=maxr;i++) {
            arr[i][minc] = oned[count];
            count++;
            if(count == n) {
                return;
            }
        }
        minc++;

        //bottom wall
        for(int j=minc;j<=maxc;j++) {
            arr[maxr][j]= oned[count];
            count++;
            if(count == n) {
                return;
            }
        }
        maxr--;

        //right wall
        for(int i=maxr;i>=minr;i--) {
            arr[i][maxc]= oned[count];
            count++;
            if(count == n) {
                return;
            }
        }
        maxc--;

        //top wall
        for(int j=maxc;j>=minc;j--) {
            arr[minr][j]= oned[count];
            count++;
            if(count == n) {
                return;
            }
        }
        minr++;
    }
}
```

```java
public static void rotate(int[] arr, int k){
// write your code here
k=k%arr.length;

if(k<0)
{
    k=k+arr.length;
}

reverse(arr,0,arr.length-k-1);
reverse(arr,arr.length-k,arr.length-1);
reverse(arr,0,arr.length-1);
}

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