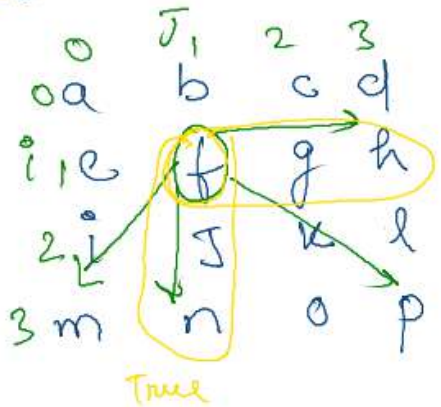
Ques Cross word



true

true
false

a.The word can be Horizontal --- From left to right.

b.The word can be Vertical - From Top to bottom.

c.The word can be along the Big-Diagonal wise - . From north-west to south-east or from north-east to south-west.

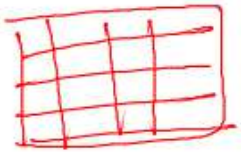
```
public static boolean isWordPresent(char[][] arr, String word){

    for(int i =0;i<arr.length;i++){
        for(int j =0;j<arr[0].length;j++){
            if(arr[i][j]==word.charAt(0)){
                boolean found = isfound(arr,word,i,j);
                if(found){
                    return true;
                }
            }
        }
    }
    return false;
}
```

$O(n^2 * O(len))$

```java
public static boolean isfound(char[][] arr,String word,int row,int col){
    int rows = arr.length;
    int cols = arr[0].length;
    int len = word.length();
    //left to right
    if(col+len<= cols){
        boolean found =true;
        for(int k=0;k<len;k++){
            if(arr[row][col+k]!= word.charAt(k)){
                found = false;
                break;
            }
        }
        if(found){
            return true;
        }
    }
    //top to down
    if(row+len<= rows){
        boolean found =true;
        for(int k=0;k<len;k++){
            if(arr[row+k][col]!= word.charAt(k)){
                found = false;
                break;
            }
        }
        if(found){
            return true;
        }
    }
    //north-west to south east
    if(row+len<=rows && col+len<=cols){
        boolean found =true;
        for(int k=0;k<len;k++){
            if(arr[row+k][col+k]!= word.charAt(k)){
                found = false;
                break;
            }
        }
        if(found){
            return true;
        }
    }
```

GREEN

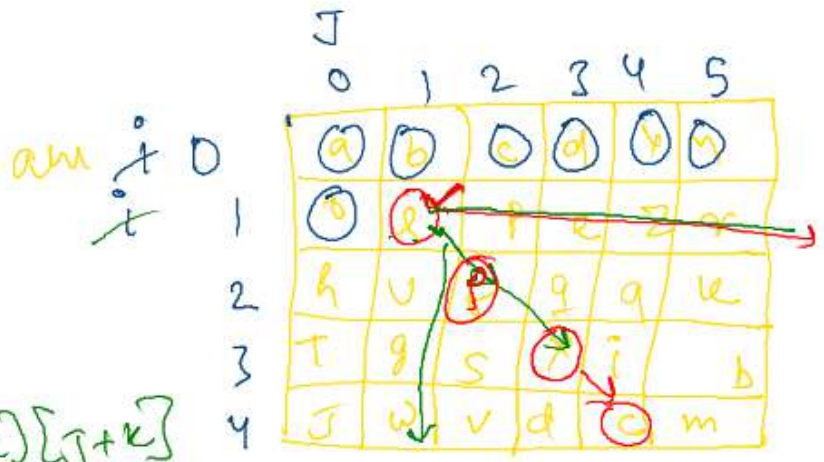
m = word.length()



$O(n^2 * m)$

wordLength

```
        //north east to south west

        if(row+len<=rows && col-len +1>=0){
            boolean found =true;
            for(int k=0;k<len;k++){
                if(arr[row+k][col-k]!= word.charAt(k)){
                    found = false;
                    break;
                }
            }
            if(found){
                return true;
            }
        }
        return false;
    }
```



isfound(arr, word, i, j);

// left to right

// top to down

// diagonally downward to right

// diagonal downward to left

arr[i+k][j+k]

if (found){
return true.

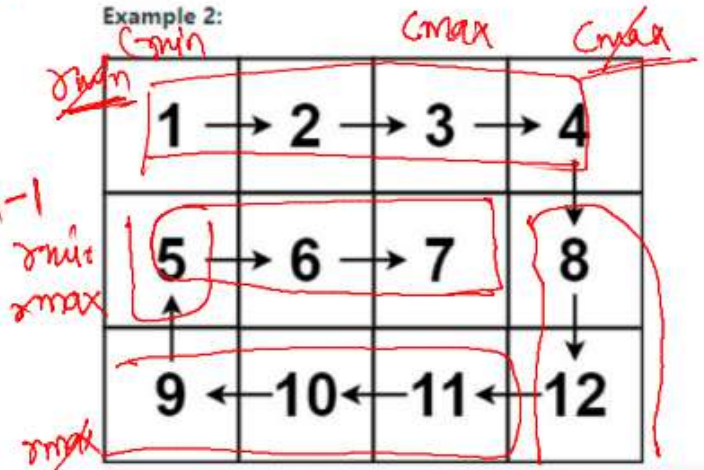
}

Qus Spiral display (imp)

rmin = 0 cmin = 0

rmax = arr.length-1 cmax = arr[0].length-1

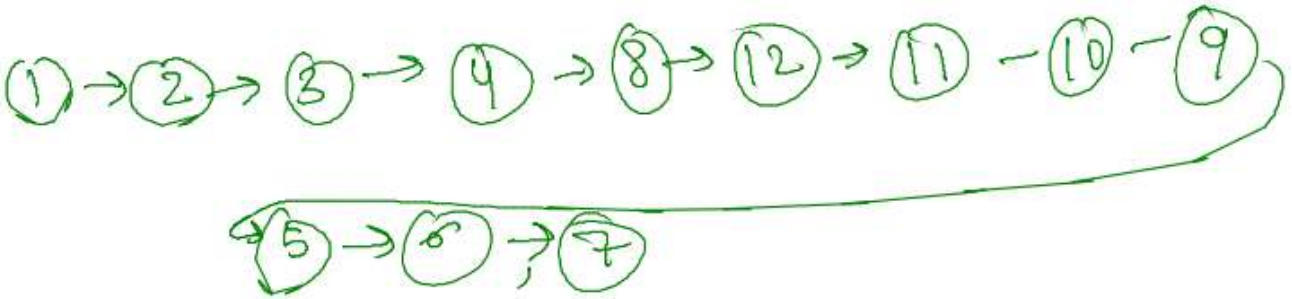
Example 2:

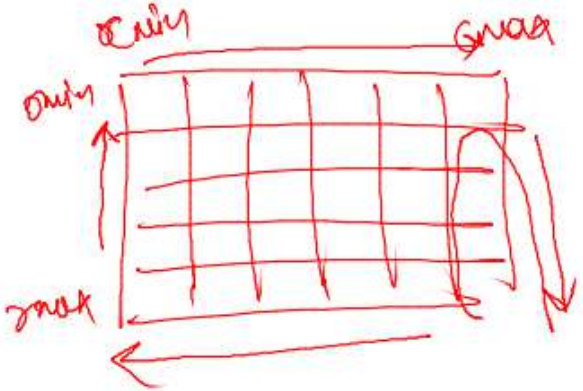


1 → 2 → 3 → 4

cmax--;

```
Input: matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
Output: [1,2,3,4,8,12,11,10,9,5,6,7]
```





cmin → cmax

rm++;

```java
public static void spiral(int[][] arr){
    int rmin=0;
    int rmax= arr.length-1;
    int cmin=0;
    int cmax= arr[0].length-1;
    int tne= arr.length*arr[0].length;
    int count=0;
    while(count<tne){
        //left to right
        for(int j =cmin;j<=cmax && count<tne;j++){
            System.out.print(arr[rmin][j]+"   ");
            count++;
        }
        rmin++;
        //top to down
        for(int i = rmin ;i<=rmax && count<tne;i++){
            System.out.print(arr[i][cmax]+"   ");
            count++;
        }
        cmax--;
        //right to left

        for(int j= cmax;j>=cmin && count<tne;j--){
            System.out.print(arr[rmax][j]+"   ");
            count++;
        }
        rmax--;
        //bottom to top
        for(int i=rmax;i>=rmin && count<tne;i--){
            System.out.print(arr[i][cmin]+"   ");
            count++;
        }
        cmin++;
    }
```