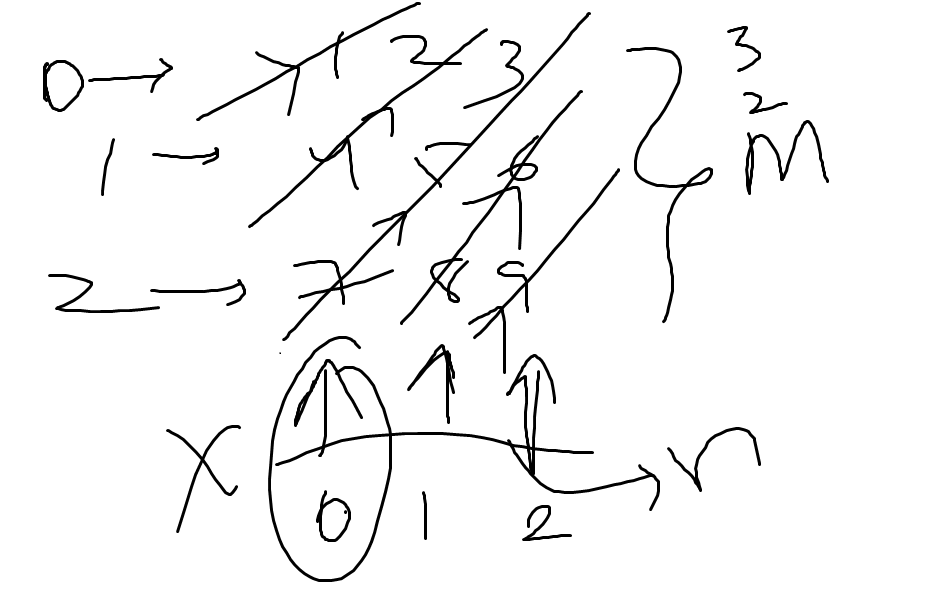
MATRIX’S - ARRAY’S - STRINGS:::

***DIAGONAL TRAVERSE OF A MATRIX:***

******

*class Solution {*

*public int[] findDiagonalOrder(int[][] mat) {*

*int idx=0;*

*int flag=0;*

*int m=mat.length;*

*int n=mat[0].length;*

*int ans[]=new int[m\*n];*

*Stack<Integer> stack=new Stack<Integer>();*

*for(int k=0;k<m;k++){*

*int i=k;*

*int j=0;*

*while(i>=0 && j<n){*

*if(flag==0){*

*ans[idx++]=mat[i][j];*

*i--;*

*j++;*

*}*

*else{*

*stack.push(mat[i][j]);*

*i--;*

*j++;*

*}*

*}*

*if(flag==0){*

*flag=1;*

*}*

*else{*

*while(!stack.isEmpty()){*

*ans[idx++]=stack.pop();*

*}*

*flag=0;*

*}*

*}*

*for(int k=1;k<n;k++){*

*int i=m-1;*

*int j=k;*

*while(j<n && i>=0){*

*if(flag==0){*

*ans[idx++]=mat[i][j];*

*i--;*

*j++;*

*}*

*else{*

*stack.push(mat[i][j]);*

*i--;*

*j++;*

*}*

*}*

*if(flag==0){*

*flag=1;*

*}*

*else{*

*while(!stack.isEmpty()){*

*ans[idx++]=stack.pop();*

*}*

*flag=0;*

*}*

*}*

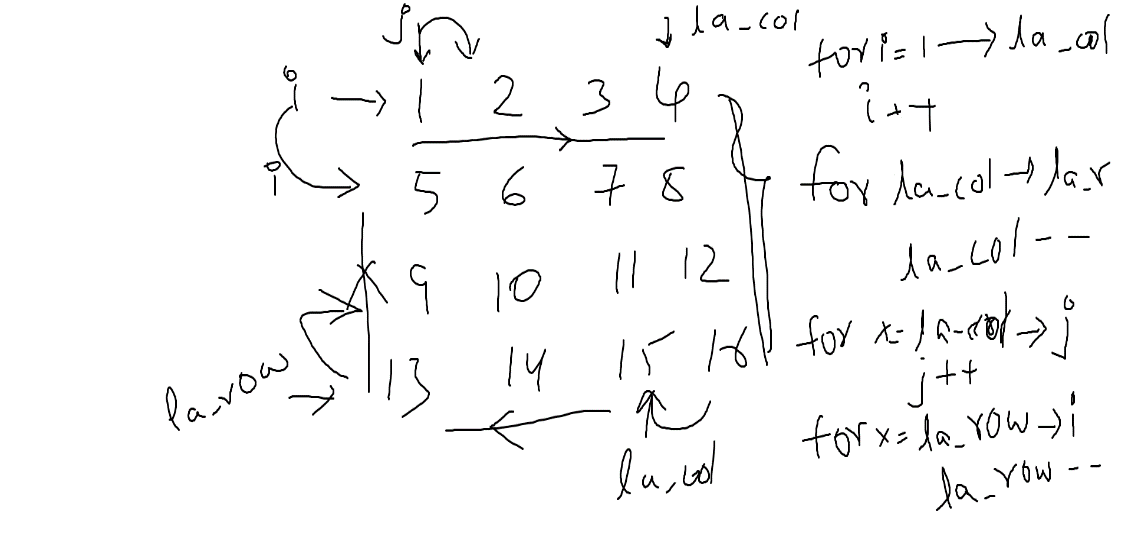
*return ans;*

*}*

*}*

*===========================================================================*

***SPIRAL TRAVERSE OF MATRIX::***



*class Solution {*

*public List<Integer> spiralOrder(int[][] matrix) {*

*List<Integer> list=new ArrayList<>();*

*int rs=0,cs=0,re=matrix.length-1,ce=matrix[0].length-1;*

*while(rs<=re && cs<=ce){*

*for(int i=cs;i<=ce;i++){*

*list.add(matrix[rs][i]);*

*}*

*rs++;*

*for(int i=rs;i<=re;i++){*

*list.add(matrix[i][ce]);*

*}*

*ce--;*

*if(ce>=cs && re>=rs){*

*for(int i=ce;i>=cs;i--){*

*list.add(matrix[re][i]);*

*}*

*re--;*

*for(int i=re;i>=rs;i--){*

*list.add(matrix[i][cs]);*

*}*

*cs++;*

*}*

*}*

*return list;*

*}*

*}*

*===========================================================================*

***PASCAL’S TRIANGLE:::***

*class Solution {*

*public List<Integer> spiralOrder(int[][] matrix) {*

*List<Integer> list=new ArrayList<>();*

*int rs=0,cs=0,re=matrix.length-1,ce=matrix[0].length-1;*

*while(rs<=re && cs<=ce){*

*for(int i=cs;i<=ce;i++){*

*list.add(matrix[rs][i]);*

*}*

*rs++;*

*for(int i=rs;i<=re;i++){*

*list.add(matrix[i][ce]);*

*}*

*ce--;*

*if(ce>=cs && re>=rs){*

*for(int i=ce;i>=cs;i--){*

*list.add(matrix[re][i]);*

*}*

*re--;*

*for(int i=re;i>=rs;i--){*

*list.add(matrix[i][cs]);*

*}*

*cs++;*

*}*

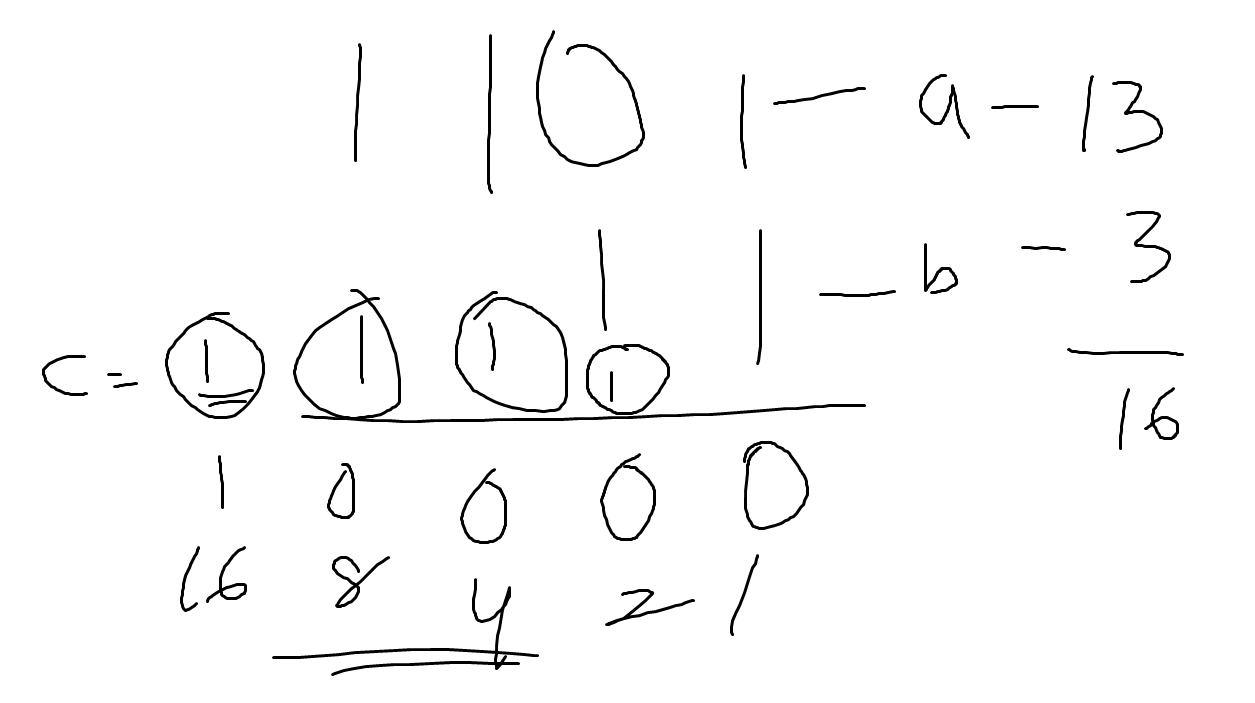
*}*

*return list;*

*}*

*}*

***ADD BINARY:::***

**

*class Solution {*

*public String addBinary(String a, String b) {*

*String ans="";*

*int i=a.length()-1;*

*int j=b.length()-1;*

*int carry=0;*

*while(i>=0 && j>=0){*

*int temp=((int)a.charAt(i)-48)+((int)b.charAt(j)-48)+carry;*

*if(temp==0){*

*ans="0"+ans;*

*carry=0;*

*}*

*else if(temp==1){*

*ans="1"+ans;*

*carry=0;*

*}*

*else if(temp==2){*

*ans="0"+ans;*

*carry=1;*

*}*

*else{*

*ans="1"+ans;*

*carry=1;*

*}*

*i--;j--;*

*}*

*while(i>=0){*

*int temp=((int)a.charAt(i)-48)+carry;*

*if(temp==0){*

*ans="0"+ans;*

*carry=0;*

*}*

*else if(temp==1){*

*ans="1"+ans;*

*carry=0;*

*}*

*else if(temp==2){*

*ans="0"+ans;*

*carry=1;*

*}*

*else{*

*ans="1"+ans;*

*carry=1;*

*}*

*i--;*

*}*

*while(j>=0){*

*int temp=((int)b.charAt(j)-48)+carry;*

*if(temp==0){*

*ans="0"+ans;*

*carry=0;*

*}*

*else if(temp==1){*

*ans="1"+ans;*

*carry=0;*

*}*

*else if(temp==2){*

*ans="0"+ans;*

*carry=1;*

*}*

*else{*

*ans="1"+ans;*

*carry=0;*

*}*

*j--;*

*}if(carry==1){*

*ans="1"+ans;*

*}*

*return ans;*

*}*

*}*

***CHECKING THE GIVEN STRING IN THE ORGINAL STRING AND RETURNING THE INDEX:::***

**Input:** haystack = "aaaaa", needle = "bba"

**Output:** -1

**Input:** haystack = "hello", needle = "ll"

**Output:** 2

*class Solution {*

*public int strStr(String haystack, String needle) {*

*int i=0;*

*while(i<=haystack.length()-needle.length()){*

*if(haystack.charAt(i)==needle.charAt(0)){*

*if(haystack.substring(i,i+needle.length()).equals(needle)){*

*return i;*

*}*

*}*

*i++;*

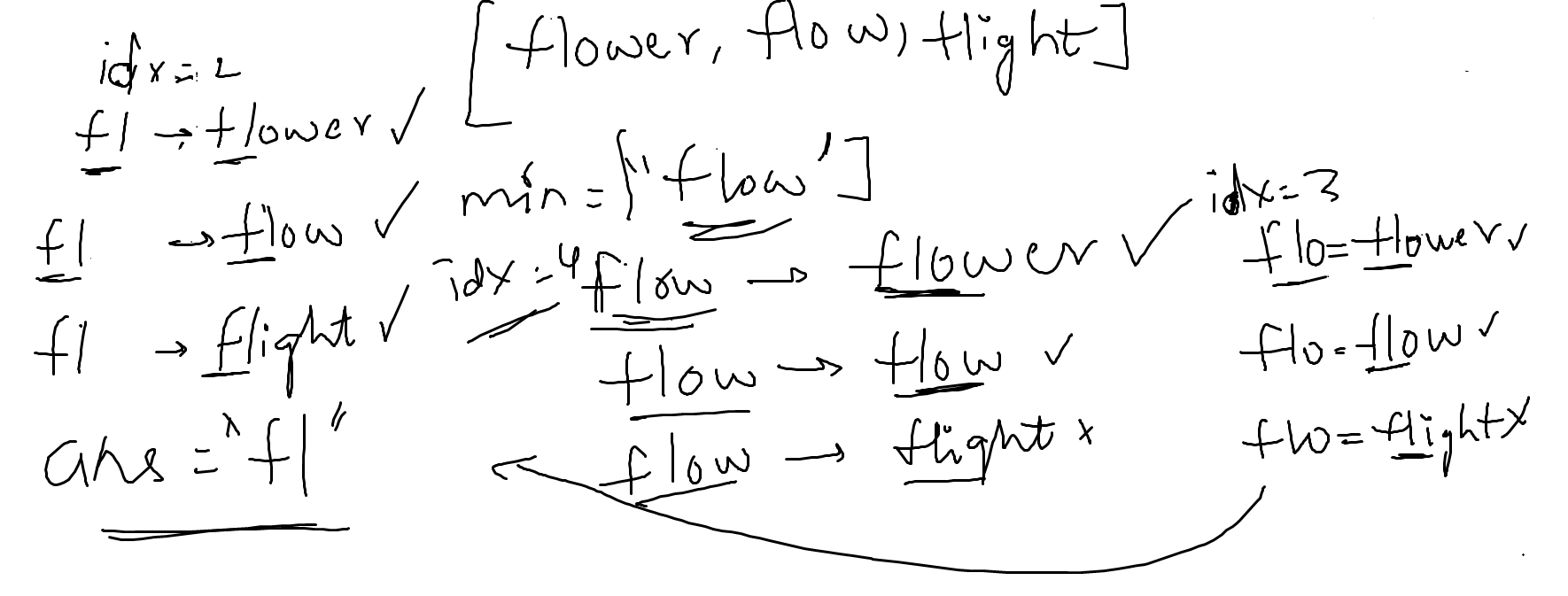
*}*

*return -1;*

*}*

*}*

***LONGEST COMMON PREFIX:::***

**

**Input:** strs = ["dog","racecar","car"]

**Output:** ""

**Explanation:** There is no common prefix among the input strings.

*class Solution {*

*public String longestCommonPrefix(String[] strs) {*

*int minidx=strs[0].length();String find=strs[0];*

*int i=0;*

*while(i<strs.length){*

*if(strs[i].length()<minidx){*

*minidx=strs[i].length();*

*find=strs[i];*

*}*

*i++;*

*}*

*String ans="";*

*int idx=find.length();int count=0;*

*while(idx>=0){*

*i=0;*

*boolean te=false;*

*while(i<strs.length){*

*if(strs[i].substring(0,idx).equals(find.substring(0,idx))){*

*te=true;*

*}*

*else{*

*te=false;*

*break;*

*}*

*i++;*

*}*

*if(te==true){*

*return find.substring(0,idx);*

*}*

*idx--;*

*}*

*return "";*

*}*

*}*

**Rotate Array**

**Input:** nums = [1,2,3,4,5,6,7], k = 3

**Output:** [5,6,7,1,2,3,4]

*class Solution {*

*public static void reverse(int arr[],int low,int high){*

*while(low<=high){*

*int temp=arr[low];*

*arr[low]=arr[high];*

*arr[high]=temp;*

*low++;*

*high--;*

*}*

*}*

*public void rotate(int[] nums, int k) {*

*int n=nums.length;*

*k=k%n;*

*reverse(nums,0,n-k-1);*

*reverse(nums,n-k,n-1);*

*reverse(nums,0,n-1);*

*}*

*}*

**Pascal's Triangle II**

**Input:** rowIndex = 3

**Output:** [1,3,3,1]

*class Solution:*

*@staticmethod*

*def fact(n:int):*

*if(n<1):*

*return 1*

*return n\*Solution.fact(n-1)*

*def getRow(self, rowIndex: int) -> List[int]:*

*stack=[]*

*k=0*

*lis=[]*

*while(k<=rowIndex/2):*

*num=Solution.fact(rowIndex)*

*denom=Solution.fact(rowIndex-k)\*Solution.fact(k)*

*c=num//denom*

*stack.append(c)*

*lis.append(c)*

*k+=1*

*if(rowIndex%2==0):*

*stack.pop()*

*while(len(stack)!=0):*

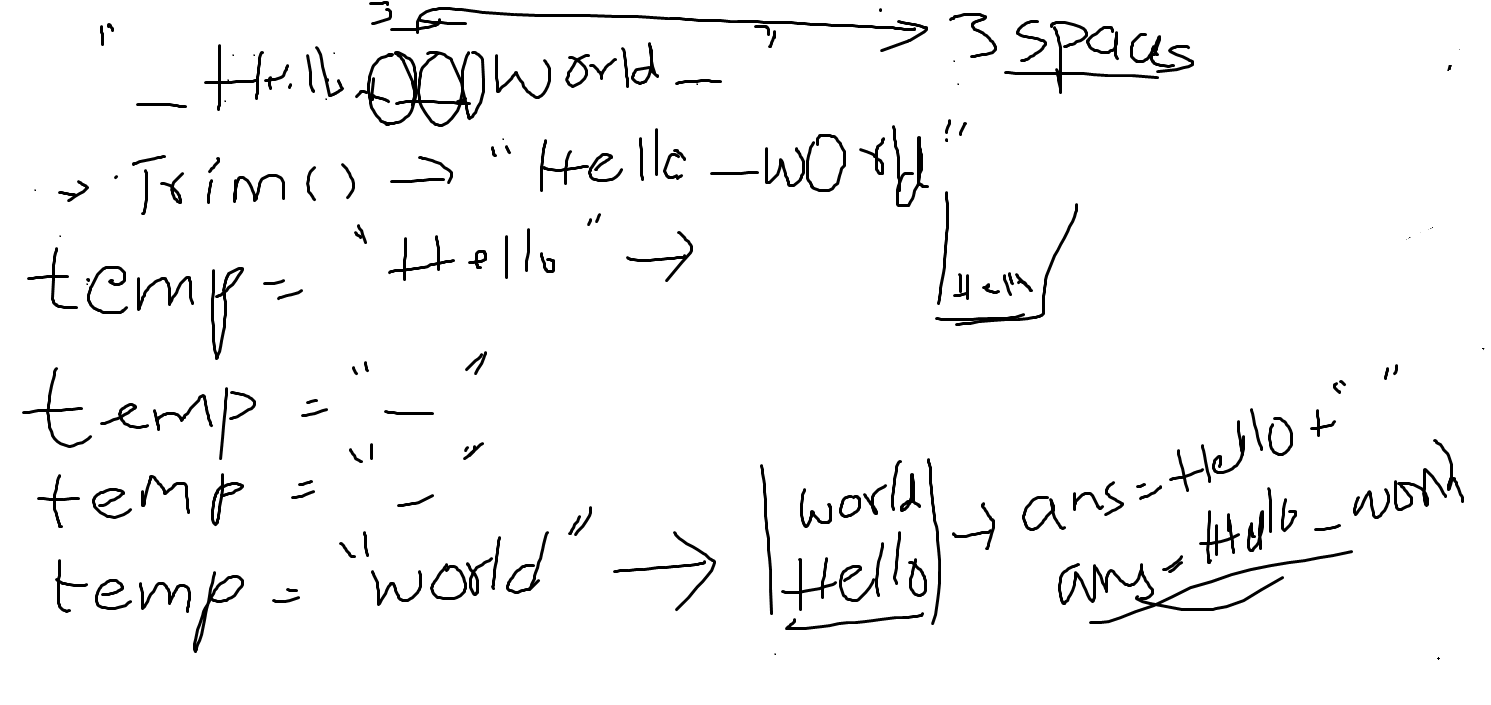
*lis.append(stack.pop())*

*return lis*

**Reverse Words in a String**

**Input:** s = "the sky is blue"

**Output:** "blue is sky the"

*class Solution {*

*public String reverseWords(String s) {*

*s=s.trim();*

*String ans="";*

*String temp="";*

*Stack<String> stack=new Stack<>();*

*for(int i=0;i<s.length();i++){*

*if(s.charAt(i)==' ' && (temp.trim()).length()!=0){*

*System.out.println(temp);*

*stack.push(temp);*

*temp="";*

*continue;*

*}*

*else if(s.charAt(i)!=' '){*

*temp+=s.charAt(i);*

*}*

*}*

*stack.push(temp);*

*int i=0;*

*while(!stack.isEmpty()){*

*ans+=stack.pop();*

*if(!stack.isEmpty()){*

*ans+=" ";*

*}*

*}*

*return ans.trim();*

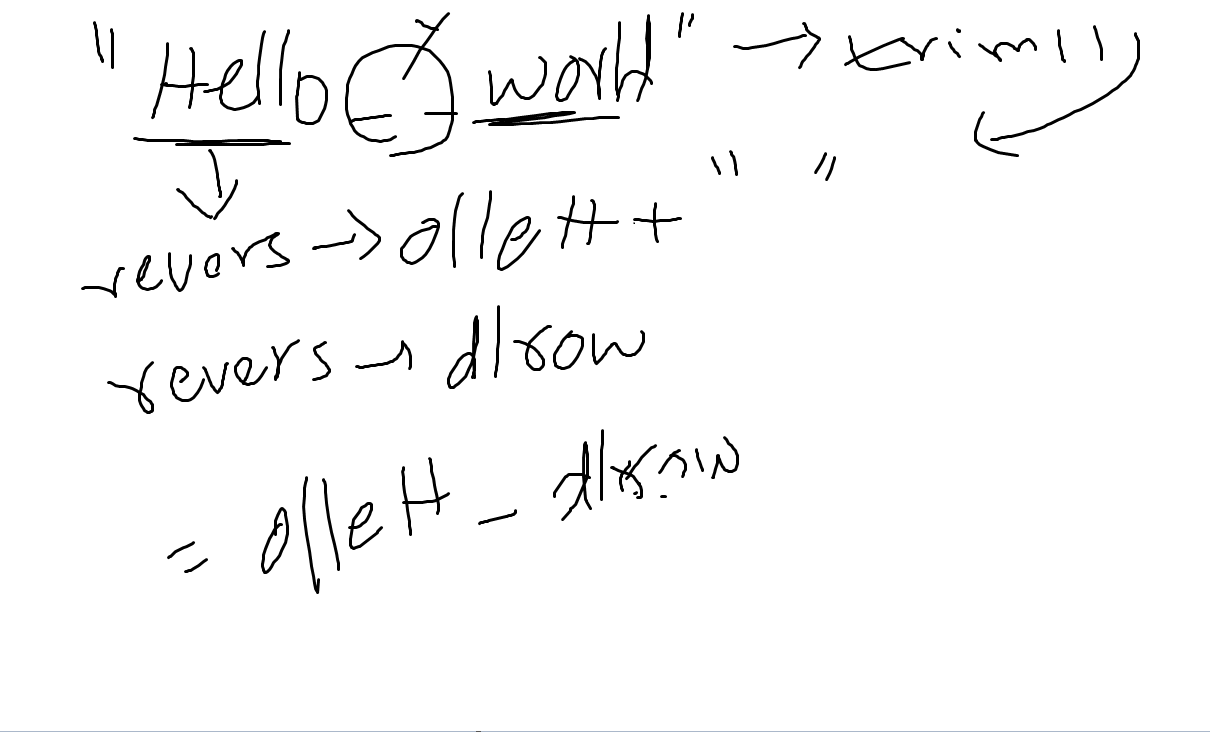
*}*

*}*

**Reverse Words in a String III**

**Input:** s = "Let's take LeetCode contest"

**Output:** "s'teL ekat edoCteeL tsetnoc"

**

class Solution {

public static String reverse(String str){

String ans="";

for(int i=str.length()-1;i>=0;i--){

if(str.charAt(i)!=' '){

ans+=str.charAt(i);

}

}

return ans;

}

public String reverseWords(String s) {

s=s.trim();

String temp="";

String ans="";

for(int i=0;i<s.length();i++){

if(s.charAt(i)==' ' && (temp.trim()).length()!=0){

ans+=reverse(temp)+" ";

temp="";

}

else if(s.charAt(i)!=' '){

temp+=s.charAt(i);

}

}

ans+=reverse(temp);

return ans;

}

}

**Remove Duplicates from Sorted Array**

**Input:** nums = [0,0,1,1,1,2,2,3,3,4]

**Output:** 5, nums = [0,1,2,3,4,\_,\_,\_,\_,\_]

**Explanation:** Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

class Solution {

public int removeDuplicates(int[] nums) {

Arrays.sort(nums);

int ans=1;

List<Integer> list=new ArrayList<>();

for(int i=0;i<nums.length-1;i++){

if(nums[i]==nums[i+1]){

continue;

}

else{

ans++;

list.add(nums[i]);

}

}

list.add(nums[nums.length-1]);

for(int i=0;i<list.size();i++){

nums[i]=list.get(i);

}

return list.size();

}

}

**Move Zeroes**

**Input:** nums = [0,1,0,3,12]

**Output:** [1,3,12,0,0]

class Solution {

public void moveZeroes(int[] nums) {

int i=0;

int j=0;

while(i<nums.length-1){

if(nums[i]==0){

j=i+1;

while(j<nums.length-1 && nums[j]==0){

j++;

}

int temp=nums[i];

nums[i]=nums[j];

nums[j]=temp;

}

i++;

}

}

}