**Question 1 Given three integer arrays arr1, arr2 and arr3 sorted in strictly increasing order, return a sorted array of only the integers that appeared in all three arrays.**

**Example 1:**

**Input: arr1 = [1,2,3,4,5], arr2 = [1,2,5,7,9], arr3 = [1,3,4,5,8]**

**Output: [1,5]**

**Prgm:**

class ThreeIntegerArray {

public static List<Integer>sortedArray(int[] arr1, int[] arr2, int[] arr3) {

List<Integer> result= new LinkedList<>()

int i = 0, j = 0, k = 0;

while (i < arr1.length && j < arr2.length && k < arr3.length) {

if (arr1[i] == arr2[j] && arr2[j] == arr3[k]) {

if (result.isEmpty() || arr1[i] !=result.get(result.size() - 1)) result.add(arr1[i]);

i++;

j++;

k++;

} else if (arr1[i] < arr2[j]) i++;

else if (arr2[j] < arr3[k]) j++;

else k++;

}

return result;

}

public static void main(String[] args) {

int arr1[] = {1,2,3,4,5};

int arr2[] = {1,2,5,7,9};

int arr3[] = {1,3,4,5,8};

System.out.println(sortedArray(arr1,arr2,arr3));

}

}

**Question 2**

**Given two 0-indexed integer arrays nums1 and nums2, return *a list* answer *of size* 2 *where:***

* **answer[0] *is a list of all distinct integers in* nums1 *which are not present in* nums2\*.\***
* **answer[1] *is a list of all distinct integers in* nums2 *which are not present in* nums1.**

**Note that the integers in the lists may be returned in any order.**

**Example 1:**

**Input: nums1 = [1,2,3], nums2 = [2,4,6]**

**Output: [[1,3],[4,6]]**

**Prgm:**

class Solution{

public static List<List<Integer>> findDifference(int[] nums1, int[] nums2) {

int i=0,j=0

Arrays.sort(nums1);

Arrays.sort(nums2);

int n=nums1.length;

int m=nums2.length;

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

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

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

while(i<n && j<m){

if(nums1[i]<nums2[j]){

if(a.size()==0 || nums1[i]!=a.get(a.size()-1))

a.add(nums1[i]);

i++;

}

else if(nums1[i]==nums2[j]){

int aa=nums1[i];

while(i<n && nums1[i]==aa)

i++;

while(j<m && nums2[j]==aa)

j++;

}

else{

if(b.size()==0 || nums2[j]!=b.get(b.size()-1))

b.add(nums2[j]);

j++;

}

}

while(i<n){

if(a.size()==0 || nums1[i]!=a.get(a.size()-1))

a.add(nums1[i]);

i++;

}

while(j<m){

if(b.size()==0 || nums2[j]!=b.get(b.size()-1))

b.add(nums2[j]);

j++;

}

ans.add(a);

ans.add(b)

return ans;

}

public static void main(String[] args) {

int nums1[] = {1,2,3};

int nums2[] = {2,4,6}

System.out.println(findDifference(nums1, nums2));

}

}

**Question 3 Given a 2D integer array matrix, return *the transpose of* matrix.**

**The transpose of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.**

**Example 1:**

**Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]**

**Output: [[1,4,7],[2,5,8],[3,6,9]]**

**Prgm:**

class Transpose\_Matrix{

public static void main(String args[]){

int a[][]={{1,2,3},{4,5,6},{7,8,9}};

int b[][] = new int[3][3];

System.out.println("Given matrix :: ");

for(int i = 0;i<3;i++){

for(int j = 0;j<3;j++){

System.out.print(a[i][j]+" ");

}

System.out.println();

}

System.out.println("Matrix after transpose :: ");

for(int i = 0;i<3;i++){

for(int j = 0;j<3;j++){

b[i][j] = 0;

for(int k = 0;k<3;k++){

b[i][j]=a[j][i];

}

System.out.print(b[i][j]+" ");

}

System.out.println();

}

}

}

**Question 4 Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2), ..., (an, bn) such that the sum of min(ai, bi) for all i is maximized. Return *the maximized sum*.**

**Example 1:**

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

**Output: 4**

**Prgm:**

class Maximized\_sum{

public static int arrayPairSum(int[] nums) {

Arrays.sort(nums);

int total = 0;

for (int i = 0; i < nums.length; i += 2) {

total += nums[i];

}

return total;

}

public static void main(String[] args) {

int nums[] = {1,4,3,2};

System.out.println(arrayPairSum(nums));

}

}

**Question 5 You have n coins and you want to build a staircase with these coins. The staircase consists of k rows where the ith row has exactly i coins. The last row of the staircase may be incomplete.**

**Given the integer n, return *the number of complete rows of the staircase you will build*.**

**Example 1:**

**Input: n = 5**

**Output: 2**

**Prgm:**

class stairCase{

public static int arrangeCoins(int n) {

if(n==0){

return 0;

}

int start = 1;

int end = n;

int mid=0;

int ans = 0;

while(start<=end){

mid = start + (end-start)/2;

if((mid\*(mid+1))/2 == n){

return mid;

}

else if((mid\*(mid+1))/2 < n){

start = mid+1;

ans = mid;

}

else{

end = mid-1;

}

}

return ans;

}

public static void main(String[] args) {

int n = 5;

System.out.println(arrangeCoins(n));

}

}

**Question 7 You are given an m x n matrix M initialized with all 0's and an array of operations ops, where ops[i] = [ai, bi] means M[x][y] should be incremented by one for all 0 <= x < ai and 0 <= y < bi.**

**Count and return *the number of maximum integers in the matrix after performing all the operations***

**Example 1:**

**Input: m = 3, n = 3, ops = [[2,2],[3,3]]**

**Output: 4**

**Prgm:**

class Solution{

public static int maxCount(int m, int n, int[][] ops) {

if(ops.length == 0){

return m \* n;

}

int row = Integer.MAX\_VALUE;

int col = Integer.MAX\_VALUE;

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

int[] temp = ops[i];

if(temp[0] < row){

row = temp[0];

}

if(temp[1] < col){

col = temp[1];

}

}

return row \* col;

}

}

**Question 8**

**Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].**

***Return the array in the form* [x1,y1,x2,y2,...,xn,yn].**

**Example 1:**

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

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

**Prgm:**

class shuffleArray {

private static int[] shuffle(int[] nums, int n) {

for(int i=n;i<2\*n;i++){

nums[i] = nums[i]<<10;

nums[i] |= nums[i-n];

}

int z = n;

for(int i=0;i<2\*n;i+=2){

nums[i] = nums[z] & 1023;

nums[i+1] = nums[z++] >> 10;

}

return nums;

}

public static void main(String[] args) {

int[] nums = {2,5,1,3,4,7};

int n = 3;

int[] output = shuffle(nums, n);

for(int i=0;i<2\*n;i++)

System.out.print(output[i]+" ");

}

}