Assignment 1 – Arrays

**Q1.** Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.

**Example:** Input: nums = [2,7,11,15], target = 9

Output0 [0,1]

Code –

import java.util.\*;

public class Adder {

    int[] arr;

    int targetSum ;

    void getData(int[] arr) {

        this.arr = arr;

    }

    void numsums(int targetSum) {

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

            for(int j=i+1; j<this.arr.length; j++) {

                if(this.arr[i] + this.arr[j] == targetSum) {

                    System.out.println("(" + i + ", "+j + ")");

                }

            }

        }

    }

    public static void main(String[] args) {

        // TODO Auto-generated method stub

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Array Size");

        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter Array Elements :");

        for(int i=0; i<n; i++) {

            int el = sc.nextInt();

            arr[i] = el;

        }

        Adder a = new Adder();

        a.getData(arr);

        System.out.println("Enter Target sum : ");

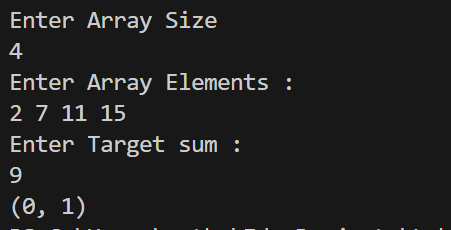
        int targetSum = sc.nextInt();

        a.numsums(targetSum);

    }

}

Output –



**Q2.** Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val.

Consider the number of elements in nums which are not equal to val be k, to get accepted, you need to do the following things:

* Change the array nums such that the first k elements of nums contain the elements which are not equal to val. The remaining elements of nums are not important as well as the size of nums.
* Return k.

**Example :** Input: nums = [3,2,2,3], val = 3 Output: 2, nums = [2,2,*\_,*\_]

Code –

import java.util.\*;

public class Q2 {

    static int k;

    static void ChangeNums(int nums[], int val){

        ArrayList<Integer> list1 = new ArrayList<>();

        ArrayList<Integer> list2 = new ArrayList<>();

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

            if(nums[i]==val){

                list2.add(nums[i]);

            }

            else{

                list1.add(nums[i]);

            }

        }

        k = list1.size();

        list1.addAll(list2);

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

            nums[i]=list1.get(i);

        }

    }

    public static void main(String[] args){

        int val;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Value");

        val=sc.nextInt();

        System.out.println("Enter Array Size");

        int n = sc.nextInt();

        int[] nums = new int[n];

        System.out.println("Enter Array Elements :");

        for(int i=0; i<n; i++) {

            nums[i] = sc.nextInt();

        }

        ChangeNums(nums,val);

        System.out.print(k+", ");

        System.out.print("[");

        for(int i=0;i<n;i++){

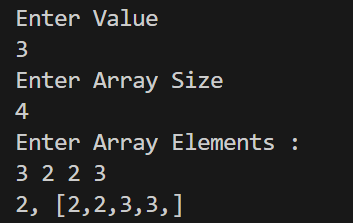
            System.out.print(nums[i]+",");

        }

        System.out.print("]");

    }

}



**Q3.** Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order. You must write an algorithm with O(log n) runtime complexity.

**Example 1:** Input: nums = [1,3,5,6], target = 5

Code –

import java.util.\*;

import java.io.\*;

class Q3\_SearchIndexPosition {

     static int SearchIndex(int nums[], int target){

        int n=nums.length;

        int start=0;

        int end=n-1;

        while(start<=end){

            int mid=(start+end)/2;

            if(nums[mid]==target){

                return mid;

            }

            else if(nums[mid]<target){

                start=mid+1;

            }

            else{

                end=mid-1;

            }

        }

        return end+1;

    }

    public static void main(String[] args) {

        int target;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Target");

        target=sc.nextInt();

        System.out.println("Enter Array Size");

        int n = sc.nextInt();

        int[] nums = new int[n];

        System.out.println("Enter Array Elements (in sorted order) :");

        for(int i=0; i<=n-1; i++) {

            nums[i] = sc.nextInt();

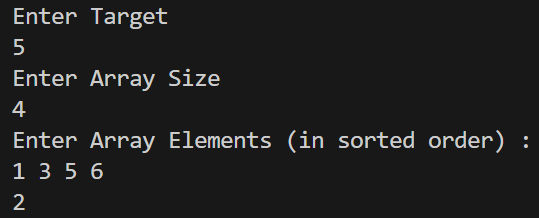
        }

        System.out.println(SearchIndex(nums,target));

    }

}

Output –



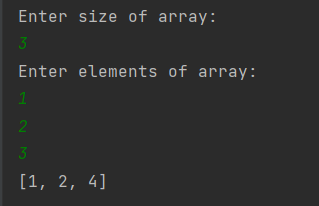
**Q4.** You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's. Increment the large integer by one and return the resulting array of digits.

**Example 1:** Input: digits = [1,2,3] Output: [1,2,4]

Code -

import java.util.\*;  
import java.io.\*;  
class Solution{  
 static int[] plusone(int arr[]){  
 int n=arr.length;  
 for(int i=n-1;i>=0;i--){  
 ++arr[i];  
 arr[i]%=10;  
 if(arr[i]!=0){  
 return arr;  
 }  
 }  
 arr = new int[n+1];  
 arr[0]=1;  
 return arr;  
 }  
  
 public static void main(String[] args) {  
 int n;  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Enter size of array: ");  
 n = sc.nextInt();  
 int[] arr = new int[n];  
 System.*out*.println("Enter elements of array: ");  
 for(int i =0; i<n; i++){  
 arr[i] = sc.nextInt();  
 }  
  
 System.*out*.println(Arrays.*toString*(*plusone*(arr)));  
 }  
}

Output –



**Q5.** You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively. Merge nums1 and nums2 into a single array sorted in non-decreasing order. The final sorted array should not be returned by the function, but instead be stored inside the array nums1. To accommodate this, nums1 has a length of m + n, where the first m elements denote the elements that should be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n.

**Example 1:** Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3

Output: [1,2,2,3,5,6]

Code –

import java.util.\*;

import java.io.\*;

class Q5\_MergeToSortedArrays {

    static void merge(int[] nums1, int m, int[] nums2, int n) {

        // nums1 = new int[m+n];

        int j=0;

        for(int i=m;i<m+n;i++){

            nums1[i] = nums2[j];

            j+=1;

        }

        Arrays.sort(nums1);

    }

    public static void main(String[] args) {

        int p;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter size of nums1: ");

        p = sc.nextInt();

        int[] nums1 = new int[p];

        System.out.println("Enter elements of nums1: ");

        for(int i =0; i<p; i++){

            nums1[i] = sc.nextInt();

        }

        System.out.println("Enter value of m: ");

        int m = sc.nextInt();

        int n;

        System.out.println("Enter size of nums2: ");

        n = sc.nextInt();

        int[] nums2 = new int[n];

        System.out.println("Enter elements of nums2: ");

        for(int i =0; i<n; i++){

            nums2[i] = sc.nextInt();

        }

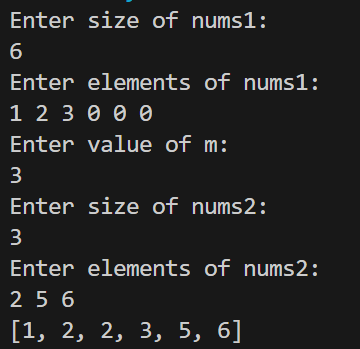
        merge(nums1,m,nums2,n);

        System.out.println(Arrays.toString(nums1));

    }

}

Output –



**Q6.** Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct.

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

Output: true

Code –

import java.util.\*;

class Q6\_ContainsDuplicate {

  static boolean containsDuplicate(int[] nums) {

    Set<Integer> seen = new HashSet<>();

    for (final int num : nums)

      if (!seen.add(num))

        return true;

    return false;

  }

  public static void main(String[] args) {

    int n;

    Scanner sc = new Scanner(System.in);

    System.out.println("Enter size of array: ");

    n = sc.nextInt();

    int[] nums = new int[n];

    System.out.println("Enter elements of array: ");

    for(int i=0; i<n; i++){

        nums[i] = sc.nextInt();

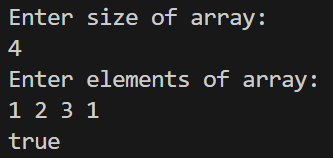
    }

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

  }

}

Output –



**Q7.** Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the nonzero elements. Note that you must do this in-place without making a copy of the array.

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

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

Code –

import java.util.\*;

class Q7\_MoveZeros{

    static void movezeros(int nums[]){

        int n = nums.length;

        int count = 0;

        for(int i = 0; i<n; i++){

            if(nums[i]!=0){

                nums[count++] = nums[i];

                // nums[count] = nums[i];

                // count++;

            }

        }

        while(count<n){

            nums[count++]=0;

            // nums[count] = 0;

            // count++;

        }

    }public static void main(String[] args) {

        int n;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter size of array: ");

        n = sc.nextInt();

        System.out.println("Enter Elements of Array: ");

        int[] nums = new int[n];

        for(int i =0; i<n; i++){

            nums[i] = sc.nextInt();

        }

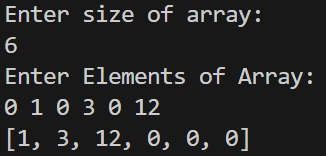
        movezeros(nums);

        System.out.println(Arrays.toString(nums));

    }

}

Output –



**Q8.** You have a set of integers s, which originally contains all the numbers from 1 to n. Unfortunately, due to some error, one of the numbers in s got duplicated to another number in the set, which results in repetition of one number and loss of another number. You are given an integer array nums representing the data status of this set after the error. Find the number that occurs twice and the number that is missing and return them in the form of an array.

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

Output: [2,3]

import java.util.\*;

import java.util.Map;

class Q8\_SetMismatch{

    public static void answer(int [] arr){

        int sumArr = 0;

        int sumNum = arr.length\*(arr.length+1)/2;

        for(int i :arr){

            sumArr +=i;

        }

        Map<Integer,Integer> freqmap = new  HashMap<Integer,Integer>();

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

            if(freqmap.containsKey(arr[i])){

                System.out.print(arr[i]+”, ”);

                System.out.println(sumNum-sumArr+arr[i]);

                break;

            }

            freqmap.put(arr[i], 1);

        }

    }

    public static void main(String[] args) {

        int n;

        System.out.println("Enter size of array: ");

        Scanner sc = new Scanner(System.in);

        n=sc.nextInt();

        int[] array = new int[n];

        System.out.println("Enter elements of array: ");

        for(int i =0; i<n; i++){

            array[i] = sc.nextInt();

        }

        answer(array);

    }

}

Output –

