1 Ans

```
public class MaxSum {
    public static int maxSum(int[] nums) {
        int maxCurrent = nums[0];
        int maxGlobal = nums[0];
        for (int i = 1; i < nums.length; i++) {
            maxCurrent = Math.max(nums[i], maxCurrent + nums[i]);
            if (maxCurrent > maxGlobal) {
                maxGlobal = maxCurrent;
            }
        }
        return maxGlobal;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        System.out.println(maxSum(arr));
    }
}
```

2 Ans

```
import java.util.Scanner;
public class MoveZeros {
   public static void moveZerosToEnd(int[] nums) {
      int n = nums.length;
      int lastNonZeroFoundAt = 0;
      for (int i = 0; i < n; i++) {
         if (nums[i] != 0) {
            int temp = nums[lastNonZeroFoundAt];
            nums[lastNonZeroFoundAt] = nums[i];
            nums[i] = temp;
            lastNonZeroFoundAt++;
         }
      }
   }
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int n = scanner.nextInt();
      int[] arr = new int[n];
      for (int i = 0; i < n; i++) {
        arr[i] = scanner.nextInt();
      }
      moveZerosToEnd(arr);
      System.out.println(Arrays.toString(arr));
      }
}
```

3 Ans

```
import java.util.Scanner;
public class MissingNumberSum {
    public static int findMissingNumber(int[] nums) {
        int n = nums.length;  (actual range is from 0 to n)
        int expectedSum = n * (n + 1) / 2;
        int actualSum = 0;
        for (int num : nums) {
            actualSum += num;
        }
        return expectedSum - actualSum;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int missingNumber = findMissingNumber(arr);
        System.out.println(missingNumber);
    }
}
```

4 Ans

```
import java.util.Scanner;
public class TwoSumBruteForce {
    public static int[] twoSum(int[] nums, int target) {
        int n = nums.length;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                if (nums[i] + nums[j] == target) {
                    return new int[] {i, j};
                }
            }
        }
        throw new IllegalArgumentException("No two sum solution");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int target = scanner.nextInt();
        int[] result = twoSum(arr, target);
        System.out.println(Arrays.toString(result));
    }
}
```

5 Ans

```
import java.util.Scanner;
import java.util.HashSet;
import java.util.ArrayList;
public class RemoveDuplicates {
    public static int[] removeDuplicates(int[] nums) {
        HashSet<Integer> seen = new HashSet<>();
        ArrayList<Integer> result = new ArrayList<>();
        for (int num : nums) {
            if (!seen.contains(num)) {
                seen.add(num);
                result.add(num);
            }
        }
        int[] uniqueArray = new int[result.size()];
        for (int i = 0; i < result.size(); i++) {
            uniqueArray[i] = result.get(i);
        }
        return uniqueArray;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
```

```
        int[] uniqueArray = removeDuplicates(arr);
        System.out.println(Arrays.toString(uniqueArray));
    }
}
```

6 Ans

```
import java.util.HashMap;
import java.util.Scanner;
public class SubarraySumEqualsK {
    public static int subarraySum(int[] nums, int k) {
        HashMap<Integer, Integer> map = new HashMap<>();
        map.put(0, 1);
        int currentSum = 0;
        int count = 0;
        for (int num : nums) {
            currentSum += num;
            if (map.containsKey(currentSum - k)) {
                count += map.get(currentSum - k);
            }
            map.put(currentSum, map.getOrDefault(currentSum, 0) + 1);
        }
        return count;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
```

```
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int k = scanner.nextInt();
        int result = subarraySum(arr, k);
        System.out.println(result);
    }
}
```

7 Ans

```
import java.util.Scanner;
public class CheckSortedArray {
    public static boolean isSorted(int[] arr) {
        for (int i = 0; i < arr.length - 1; i++) {
            if (arr[i] > arr[i + 1]) {
                return false;
            }
        }
        return true;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
```

```
        boolean ascending = isSorted(arr);
        if (ascending) {
            System.out.println("1");
        } else {
            System.out.println("0");
        }
    }
}
```

8 Ans

```
import java.util.Scanner;
public class SortColors {
    public static void sortColors(int[] nums) {
        int low = 0, mid = 0, high = nums.length - 1;
        while (mid <= high) {
            if (nums[mid] == 0) {
                swap(nums, low, mid);
                low++;
                mid++;
            } else if (nums[mid] == 1) {
                mid++;
            } else {
                swap(nums, mid, high);
                high--;
            }
        }
    }
```

```
    private static void swap(int[] nums, int i, int j) {
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        sortColors(arr);
        System.out.println(Arrays.toString(arr));
    }
}
```

9 Ans

```
import java.util.Scanner;

public class Solution {
    public int findMaxConsecutiveOnes(int[] nums) {
        int count = 0;
        int maxCount = 0;
        for (int num : nums) {
            if (num == 1) {
                count++;
```

```
            } else {
                if (count > maxCount) {
                    maxCount = count;
                }
                count = 0;
            }
        }
        if (count > maxCount) {
            maxCount = count;
        }
        return maxCount;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        Solution solution = new Solution();
        int result = solution.findMaxConsecutiveOnes(nums);
        System.out.println(result);
    }
}
```

10 Ans

```
import java.util.HashMap;
import java.util.Scanner;

public class Solution {
    public int singleNumber(int[] nums) {
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int num : nums) {
            if (map.containsKey(num)) {
                map.put(num, map.get(num) + 1);
            } else {
                map.put(num, 1);
            }
        }
        for (int num : map.keySet()) {
            if (map.get(num) == 1) {
                return num;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
```

```
            nums[i] = scanner.nextInt();
        }
        Solution solution = new Solution();
        int result = solution.singleNumber(nums);
        System.out.println( result);
    }
}
```

11 Ans

```
import java.util.Scanner;
public class Solution {
    public int longestSubarrayWithSumK(int[] arr, int K) {
        int start = 0;
        int end = 0;
        int currentSum = 0;
        int maxLength = 0;
        while (end < arr.length) {
            currentSum += arr[end];
            while (currentSum > K && start <= end) {
                currentSum -= arr[start];
                start++;
            }
            if (currentSum == K) {
                maxLength = Math.max(maxLength, end - start + 1);
            }
            end++;
        }
        return maxLength;
```

```
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int K = scanner.nextInt();
        Solution solution = new Solution();
        int result = solution.longestSubarrayWithSumK(arr, K);
        System.out.println(result);
    }
}
```

12 Ans

```
import java.util.Scanner;

public class Solution {
    public static int[] mergeSortedArrays(int[] arr1, int[] arr2) {
        int n1 = arr1.length;
        int n2 = arr2.length;
        int[] mergedArray = new int[n1 + n2];
        int i = 0, j = 0, k = 0;
        while (i < n1 && j < n2) {
            if (arr1[i] <= arr2[j]) {
```

```
            mergedArray[k++] = arr1[i++];
        } else {
            mergedArray[k++] = arr2[j++];
        }
    }
    while (i < n1) {
        mergedArray[k++] = arr1[i++];
    }
    while (j < n2) {
        mergedArray[k++] = arr2[j++];
    }
    return mergedArray;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int n1 = scanner.nextInt();
    int n2 = scanner.nextInt();
    int[] arr1 = new int[n1];
    int[] arr2 = new int[n2];
    for (int i = 0; i < n1; i++) {
        arr1[i] = scanner.nextInt();
    }
    for (int i = 0; i < n2; i++) {
        arr2[i] = scanner.nextInt();
    }
    int[] result = mergeSortedArrays(arr1, arr2);
```

```
        System.out.println("Merged Sorted Array: ");
        for (int num : result) {
            System.out.print(num + " ");
        }
    }
}
```

13 Ans

```
java.util.Arrays;
import java.util.Scanner;

public class RotateArrayExtraSpace {
    public static void rotate(int[] arr, int k) {
        int n = arr.length;
        k = k % n;
        int[] rotatedArray = new int[n];
        for (int i = 0; i < n; i++) {
            rotatedArray[(i + k) % n] = arr[i];
        }
        for (int i = 0; i < n; i++) {
            arr[i] = rotatedArray[i];
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
```

```
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int k = scanner.nextInt();
        rotate(arr, k);
        System.out.println(Arrays.toString(arr));
    }
}
```

14 Ans

```
import java.util.*;

public class RemoveDuplicates {
    public static int removeDuplicates(int[] nums) {
        HashSet<Integer> uniqueElements = new HashSet<>();
        for (int num : nums) {
            uniqueElements.add(num);
        }
        int index = 0;
        for (int num : uniqueElements) {
            nums[index++] = num;
        }
        return uniqueElements.size();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
```

```
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        int newLength = removeDuplicates(nums);
        System.out.print(newLength+", nums="+Arrays.toString(nums));
    }
}
```

15 Ans

```
import java.util.*;

public class RearrangeArrayBySign {
    public static int[] rearrange(int[] nums) {
        List<Integer> positives = new ArrayList<>();
        List<Integer> negatives = new ArrayList<>();
        for (int num : nums) {
            if (num > 0) {
                positives.add(num);
            } else {
                negatives.add(num);
            }
        }
        int[] rearranged = new int[nums.length];
        int index = 0;
        for (int i = 0; i < positives.size(); i++) {
            rearranged[index++] = positives.get(i);
```

```
            rearranged[index++] = negatives.get(i);
        }
        return rearranged;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        int[] rearrangedArray = rearrange(nums);
        System.out.println("Rearranged Array: ");
        for (int num : rearrangedArray) {
            System.out.print(num + " ");
        }
    }
}
```

16 Ans

```
import java.util.Scanner;

public class BestTimeToBuyAndSellStock {
    public static int maxProfit(int[] prices) {
        int minPrice = Integer.MAX_VALUE;
        int maxProfit = 0;
```

```
        for (int price : prices) {
            if (price < minPrice) {
                minPrice = price;
            }
            int profit = price - minPrice;
            if (profit > maxProfit) {
                maxProfit = profit;
            }
        }
        return maxProfit;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] prices = new int[n];
        for (int i = 0; i < n; i++) {
            prices[i] = scanner.nextInt();
        }
        int maxProfit = maxProfit(prices);
        System.out.println(maxProfit);
    }
}
```

17 Ans

```
import java.util.*;

public class ThreeSum {
    public static List<List<Integer>> threeSum(int[] nums) {
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(nums);
        for (int i = 0; i < nums.length - 2; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }
            int left = i + 1;
            int right = nums.length - 1;
            while (left < right) {
                int sum = nums[i] + nums[left] + nums[right];
                if (sum == 0) {
                    result.add(Arrays.asList(nums[i], nums[left], nums[right]));
                    left++;
                    right--;
                    while (left < right && nums[left] == nums[left - 1]) {
                        left++;
                    }
                    while (left < right && nums[right] == nums[right + 1]) {
                        right--;
                    }
                } else if (sum < 0) {
                    left++;
```

```
            } else {
                right--;
            }
        }
    }

    return result;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int n = scanner.nextInt();
    int[] nums = new int[n];
    for (int i = 0; i < n; i++) {
        nums[i] = scanner.nextInt();
    }
    List<List<Integer>> triplets = threeSum(nums);
    System.out.println( triplets);
}
}
```

18 Ans

```
import java.util.*;

public class FourSum {
    public static List<List<Integer>> fourSum(int[] nums, int target) {
        List<List<Integer>> result = new ArrayList<>();
```

```
        Arrays.sort(nums);
        for (int i = 0; i < nums.length - 3; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }
            for (int j = i + 1; j < nums.length - 2; j++) {
                if (j > i + 1 && nums[j] == nums[j - 1]) {
                    continue;
                }
                int left = j + 1;
                int right = nums.length - 1;
                while (left < right) {
                    int sum = nums[i] + nums[j] + nums[left] + nums[right];

                    if (sum == target) {
                        result.add(Arrays.asList(nums[i], nums[j], nums[left],
nums[right]));
                        left++;
                        right--;
                        while (left < right && nums[left] == nums[left - 1]) {
                            left++;
                        }
                        while (left < right && nums[right] == nums[right + 1]) {
                            right--;
                        }
                    } else if (sum < target) {
                        left++;
                    } else {
```

```
                    right--;
                }
            }
        }
    }
    return result;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int n = scanner.nextInt();
    int[] nums = new int[n];
    for (int i = 0; i < n; i++) {
        nums[i] = scanner.nextInt();
    }
    int target = scanner.nextInt();
    List<List<Integer>> quadruplets = fourSum(nums, target);
    System.out.println(quadruplets);
}
}
```

19 Ans

```
import java.util.*;

public class LongestSubarrayWithZeroSum {
    public static int maxLengthZeroSum(int[] arr) {
        HashMap<Integer, Integer> sumIndexMap = new HashMap<>();
        int maxLength = 0;
        int cumulativeSum = 0;
        for (int i = 0; i < arr.length; i++) {
            cumulativeSum += arr[i];
            if (cumulativeSum == 0) {
                maxLength = i + 1;
            }
            if (sumIndexMap.containsKey(cumulativeSum)) {
                int length = i - sumIndexMap.get(cumulativeSum);
                maxLength = Math.max(maxLength, length);
            } else {
                sumIndexMap.put(cumulativeSum, i);
            }
        }
        return maxLength;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
```

```
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int maxLength = maxLengthZeroSum(arr);
        System.out.println(maxLength);
    }
}
```

20 Ans

```
import java.util.*;

public class CountSubarraysWithGivenXor {
    public static int countSubarraysWithXorK(int[] arr, int K) {
        HashMap<Integer, Integer> xorCountMap = new HashMap<>();
        int cumulativeXor = 0;
        int count = 0;

        for (int num : arr) {
            cumulativeXor ^= num;
            if (cumulativeXor == K) {
                count++;
            }

            if (xorCountMap.containsKey(cumulativeXor ^ K)) {
                count += xorCountMap.get(cumulativeXor ^ K);
            }
```

```
xorCountMap.put(cumulativeXor,xorCountMap.getOrDefault(cumulativeXor,
0) + 1);
        }
        return count;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        int K = scanner.nextInt();
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int count = countSubarraysWithXorK(arr, K);
        System.out.println(count);
    }
}
```

21 Ans

```
import java.util.Scanner;

public class MajorityElement {
    public static int majorityElement(int[] nums) {
        int candidate = nums[0];
        int count = 1;
        for (int i = 1; i < nums.length; i++) {
```

```
            if (nums[i] == candidate) {
                count++;
            } else {
                count--;
            }
            if (count == 0) {
                candidate = nums[i];
                count = 1;
            }
        }
        return candidate
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        int majority = majorityElement(nums);
        System.out.println(majority);
    }
}
```

22 Ans

```
import java.util.Scanner;


public class SecondLargestAndSmallest {
   public static int[] findSecondLargestAndSmallest(int[] arr) {
      int n = arr.length;
      if (n < 2) {
         throw new IllegalArgumentException("Array must have at least two
elements.");
      }
      int largest = Integer.MIN_VALUE;
      int secondLargest = Integer.MIN_VALUE;
      int smallest = Integer.MAX_VALUE;
      int secondSmallest = Integer.MAX_VALUE;
      for (int num : arr) {
         if (num > largest) {
            secondLargest = largest;
            largest = num;
         } else if (num > secondLargest) {
            secondLargest = num;
         }
         if (num < smallest) {
            secondSmallest = smallest;
            smallest = num;
         } else if (num < secondSmallest) {
            secondSmallest = num;
         }
      }
```

```
        return new int[]{secondLargest, secondSmallest};
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int[] result = findSecondLargestAndSmallest(arr);
        System.out.println("Second Largest: " + result[0]);
        System.out.println("Second Smallest: " + result[1]);
    }
}
```

23 Ans

Refer 13th Answer.

24 Ans

```
import java.util.Scanner;

public class FindFirstAndLastPosition {
    public static int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};
        result[0] = findLeftPosition(nums, target);
        result[1] = findRightPosition(nums, target);
```

```
    return result;
}
private static int findLeftPosition(int[] nums, int target) {
    int left = 0, right = nums.length - 1, leftIndex = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target) {
            leftIndex = mid;
            right = mid - 1;
        } else if (nums[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return leftIndex;
}


private static int findRightPosition(int[] nums, int target) {
    int left = 0, right = nums.length - 1, rightIndex = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target) {
            rightIndex = mid;
            left = mid + 1;
        } else if (nums[mid] < target) {
```

```
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return rightIndex;
  }
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int n = scanner.nextInt();
    int[] nums = new int[n];
    for (int i = 0; i < n; i++) {
      nums[i] = scanner.nextInt();
    }
    int target = scanner.nextInt();
    int[] result = searchRange(nums, target);
    System.out.println(" [" + result[0] + ", " + result[1] + "]");
  }
}
```

25 Ans

```
import java.util.*;

public class MonkAndStudents {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int T = scanner.nextInt();
```

```
        for (int t = 0; t < T; t++) {
            int N = scanner.nextInt();
            int M = scanner.nextInt();
            HashSet<Long> candySet = new HashSet<>();
            for (int i = 0; i < N; i++) {
                long candy = scanner.nextLong();
                candySet.add(candy);
            }
            for (int i = 0; i < M; i++) {
                long newCandy = scanner.nextLong();
                if (candySet.contains(newCandy)) {
                    System.out.println("YES");
                } else {
                    System.out.println("NO");
                }
            }
        }
    }
}
```