# Q1. <https://leetcode.com/problems/two-sum/>

## Solution:

import java.util.Arrays;  
import java.util.HashMap;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 22/05/2023  
 \*/*public class QuestionOneSolution {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*findTwoSum*(new int[]{2, 3, 17, 5, 8, 11, 9, 7}, 9)));  
 System.*out*.println(Arrays.*toString*(*findTwoSum*(new int[]{2, 3, 17, 5, 8, 11, 7, 9}, 15)));  
 }  
  
 static int[] findTwoSum(int[] nums, int target) {  
 HashMap<Integer, Integer> map = new HashMap<>();  
 for (int i = 0; i < nums.length; i++) {  
 int diff = target - nums[i];  
 if (map.containsKey(diff)) {  
 return new int[]{map.get(diff), i};  
 }  
 map.put(nums[i], i);  
 }  
 return null;  
 }  
}

# Q2. <https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/>

## Solution*:*

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 22/05/2023  
 \*/*public class QuestionTwoSolution {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*findTwoSumOfSortedArray*(new int[]{2, 5, 7, 9, 11}, 9)));  
 System.*out*.println(Arrays.*toString*(*findTwoSumOfSortedArray*(new int[]{2, 4, 5, 8, 9, 11}, 15)));  
 }  
  
 static int[] findTwoSumOfSortedArray(int[] numbers, int target) {  
 int i = 0;  
 int j = numbers.length - 1;  
 while (i < j) {  
 if (numbers[i] + numbers[j] == target) {  
 return new int[]{i + 1, j + 1};  
 } else if (numbers[i] + numbers[j] < target) {  
 i++;  
 } else if (numbers[i] + numbers[j] > target) {  
 j--;  
 }  
 }  
 return null;  
 }  
}

# Q3. <https://leetcode.com/problems/merge-sorted-array/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 22/05/2023  
 \*/*public class QuestionThreeSolution {  
 public static void main(String[] args) {  
 *merge*(new int[]{1, 2, 3, 0, 0, 0}, 3, new int[]{2, 5, 6}, 3);  
 *merge*(new int[]{1}, 1, new int[]{}, 0);  
 *merge*(new int[]{0}, 0, new int[]{1}, 1);  
 }  
  
 public static void merge(int[] nums1, int m, int[] nums2, int n) {  
 int i = m - 1;  
 int j = n - 1;  
 int k = m + n - 1;  
 while (j >= 0) {  
 if (i >= 0 && nums1[i] > nums2[j]) {  
 nums1[k--] = nums1[i--];  
 } else {  
 nums1[k--] = nums2[j--];  
 }  
 }  
 System.*out*.println(Arrays.*toString*(nums1));  
 }  
}

# Q4. <https://leetcode.com/problems/pascals-triangle/>

## Solution:

import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionFourSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*pascalTriangle*(4));  
 System.*out*.println(*pascalTriangle*(6));  
 }  
  
 static List<List<Integer>> pascalTriangle(int numRows) {  
 List<List<Integer>> result = new ArrayList<>();  
 for (int i = 0; i <= numRows; i++) {  
 List<Integer> row = new ArrayList<>();  
 row.add(1);  
 for (int j = 1; j < i; j++) {  
 row.add(result.get(i - 1).get(j)  
 + result.get(i - 1).get(j - 1)  
 );  
 }  
 if (i > 0) row.add(1);  
 result.add(row);  
 }  
 return result;  
 }  
}

# Q5. <https://leetcode.com/problems/pascals-triangle-ii/>

## Solution:

import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionFiveSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*pascalTriangleFindRow*(1));  
 System.*out*.println(*pascalTriangleFindRow*(3));  
 System.*out*.println(*pascalTriangleFindRow*(0));  
 System.*out*.println(*generateNthRow*(30));  
 }  
  
 static List<Integer> pascalTriangleFindRow(int rowIndex) {  
 List<List<Integer>> result = new ArrayList<>();  
 for (int i = 0; i < rowIndex + 1; i++) {  
 List<Integer> row = new ArrayList<>();  
 row.add(1);  
 for (int j = 1; j < i; j++) {  
 row.add(result.get(i - 1).get(j)  
 + result.get(i - 1).get(j - 1)  
 );  
 }  
 if (i > 0) row.add(1);  
 result.add(row);  
 }  
 return result.get(rowIndex);  
 }  
  
 */\* using NC0, NC1, ......, NCN - 1, NCN  
 \* NCr = NC(r - 1) \* (N - r + 1)) / r  
 \*/* static List<Integer> generateNthRow(int rowIndex) {  
 int prevElement = 1;  
 List<Integer> result = new ArrayList<>();  
 result.add(prevElement);  
 if (rowIndex == 0) return result;  
 for (int i = 1; i <= rowIndex; i++) {  
 long currElement = ((long) prevElement \* (rowIndex - i + 1) / i);  
 prevElement = (int) currElement;  
 result.add(prevElement);  
 }  
 return result;  
 }  
}

# Q6. <https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionSixSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findProfit*(new int[]{7, 1, 5, 3, 6, 4}));  
 System.*out*.println(*findProfit*(new int[]{7, 6, 4, 3, 1}));  
 }  
  
 static int findProfit(int[] prices) {  
 int profit = 0;  
 int min = Integer.*MAX\_VALUE*;  
 for (int i = 0; i < prices.length; i++) {  
 min = Math.*min*(min, prices[i]);  
 profit = Math.*max*(profit, prices[i] - min);  
 }  
 return profit;  
 }  
}

# Q7. <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionSevenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findProfit*(new int[]{7, 1, 5, 3, 6, 4}));  
 System.*out*.println(*findProfit*(new int[]{7, 6, 4, 3, 1}));  
 }  
  
 static int findProfit(int[] prices) {  
 int profit = 0;  
 int min = Integer.*MAX\_VALUE*;  
 for (int i = 0; i < prices.length; i++) {  
 min = Math.*min*(min, prices[i]);  
 if (prices[i] - min > 0) {  
 profit += prices[i] - min;  
 min = prices[i];  
 }  
 }  
 return profit;  
 }  
}

# Q8. <https://leetcode.com/problems/majority-element/>

## Solution:

import java.util.Arrays;  
import java.util.HashMap;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionEightSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findMajorityElementMethod1*(new int[1]));  
 System.*out*.println(*findMejorityElementMethod2*(new int[]{2,2,1,1,1,2,2}));  
 }  
  
 *//TC -> O(n), SC -> O(n)* static int findMajorityElementMethod1(int[] nums) {  
 HashMap<Integer, Integer> map = new HashMap<>();  
 for (int i = 0; i < nums.length; i++) {  
 if (map.containsKey(nums[i])) {  
 int count = map.get(nums[i]) + 1;  
 if (count > nums.length / 2) {  
 return nums[i];  
 } else {  
 map.put(nums[i], count);  
 }  
 } else {  
 map.put(nums[i], 1);  
 }  
 }  
 return 1;  
 }  
  
 *//TC -> O(nLogn), SC -> O(1)* static int findMejorityElementMethod2(int[] nums) {  
 Arrays.*sort*(nums);  
 return nums[nums.length / 2];  
 }  
}

# Q9. <https://leetcode.com/problems/majority-element-ii/>

## Solution:

import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionNineSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findMejorityElements*(new int[]{3, 2, 3}));  
 }  
  
 static List<Integer> findMejorityElements(int[] nums) {  
 HashMap<Integer, Integer> map = new HashMap<>();  
 for (int num : nums) {  
 if (map.containsKey(num)) {  
 int count = map.get(num) + 1;  
 map.put(num, count);  
 } else {  
 map.put(num, 1);  
 }  
 }  
 List<Integer> list = new ArrayList<>();  
 for (Map.Entry<Integer, Integer> entry : map.entrySet()) {  
 int value = entry.getValue();  
 int key = entry.getKey();  
 if (value > nums.length / 3) {  
 list.add(key);  
 }  
 }  
 return list;  
 }  
}

# Q10. <https://leetcode.com/problems/missing-ranges/>

## Solution:

import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionTenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findMissingElements*(new int[]{3, 1, 2, 4, 10}, 2, 10));  
 }  
  
 static List<Integer> findMissingElements(int[] nums, int low, int high) {  
 HashSet<Integer> set = new HashSet<>();  
 List<Integer> list = new ArrayList<>();  
 for (int num : nums) {  
 set.add(num);  
 }  
 for (int i = low; i <= high; i++) {  
 if (!set.contains(i)) {  
 list.add(i);  
 }  
 }  
 return list;  
 }  
}

# Q11. <https://leetcode.com/problems/3sum/>

## Solution:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.HashSet;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 24/05/2023  
 \*/*public class QuestionElevenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findThreeSum*(new int[]{-1, 0, 1, 2, -1, -4}));  
 }  
  
 static List<List<Integer>> findThreeSum(int[] nums) {  
 HashSet<List<Integer>> result = new HashSet<>();  
 if (nums.length == 0) return new ArrayList<>(result);  
 Arrays.*sort*(nums);  
 for (int i = 0; i < nums.length - 2; i++) {  
 int j = i + 1;  
 int k = nums.length - 1;  
 while (j < k) {  
 int sum = nums[j] + nums[k];  
 if (sum == -nums[i]) {  
 result.add(Arrays.*asList*(nums[i], nums[j], nums[k]));  
 j++;  
 k--;  
 } else if (sum > -nums[i]) k--;  
 else if (sum < -nums[i]) j++;  
 }  
 }  
 return new ArrayList<>(result);  
 }  
}

# Q13. <https://leetcode.com/problems/3sum-closest/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 24/05/2023  
 \*/*public class QuestionThirteenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*threeSumClosest*(new int[]{-1, 2, 1, -4},1));  
 }  
  
 static int threeSumClosest(int[] nums, int target) {  
 int min = Integer.*MAX\_VALUE*;  
 int result = 0;  
 Arrays.*sort*(nums);  
 for (int i = 0; i < nums.length - 2; i++) {  
 int j = i + 1;  
 int k = nums.length - 1;  
 while (j < k) {  
 int sum = nums[i] + nums[j] + nums[k];  
 int diff = Math.*abs*(sum - target);  
 if (diff == 0) return sum;  
 if (diff < min) {  
 min = diff;  
 result = sum;  
 }  
 if (sum < target) j++;  
 else k--;  
 }  
 }  
 return result;  
 }  
}

# Q14. <https://leetcode.com/problems/4sum/>

## Solution:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.HashSet;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 27/05/2023  
 \*/*public class QuestionFourteenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*findFourSum*(new int[]{1, 0, -1, 0, -2, 2}, 0));  
 System.*out*.println(*findFourSum*(new int[]{-1, 0, 1, 2, -1, -4}, -1));  
 }  
  
 static List<List<Integer>> findFourSum(int[] nums, int target) {  
 List<List<Integer>> result = new ArrayList<>();  
 int n = nums.length;  
 Arrays.*sort*(nums);  
 for (int i = 0; i < n - 2; i++) {  
 if (i > 0 && nums[i] == nums[i - 1]) {  
 continue;  
 }  
 for (int j = i + 1; j < n - 2; j++) {  
 if (j > i + 1 && nums[j] == nums[j - 1]) {  
 continue;  
 }  
 int left = j + 1;  
 int right = n - 1;  
 while (left < right) {  
 long sum = (long) nums[i] + nums[j] + nums[left] + nums[right];  
 if (sum == target) {  
 result.add(Arrays.*asList*(nums[i], nums[j], nums[left], nums[right]));  
 while (left < right && nums[left] == nums[left + 1]) left++;  
 while (left < right && nums[right] == nums[right - 1]) right--;  
 left++;  
 right--;  
 } else if (sum < target) {  
 left++;  
 } else if (sum > target) {  
 right--;  
 }  
 }  
 }  
 }  
 return new ArrayList<>(result);  
 }  
}

# Q15. <https://leetcode.com/problems/rotate-image/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 23/05/2023  
 \*/*public class QuestionFifteenSolution {  
 public static void main(String[] args) {  
 *rotate*(new int[][]{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}});  
 }  
  
 static void rotate(int[][] matrix) {  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = i; j < matrix.length; j++) {  
 int temp = matrix[i][j];  
 matrix[i][j] = matrix[j][i];  
 matrix[j][i] = temp;  
 }  
 }  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix.length/2; j++) {  
 int temp = matrix[i][j];  
 matrix[i][j] = matrix[i][matrix.length - 1 - j];  
 matrix[i][matrix.length - 1 - j] = temp;  
 }  
 }  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
}