# Q1. Reverse the Number like 153 => 351. Numbers can be negative.

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 16/05/2023  
 \*/*public class QuestionOneSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*reverseNumber*(153));  
 System.*out*.println(*reverseNumber*(-4861));  
 System.*out*.println(*reverseNumber*(-84959100));  
 }  
  
 *//method to reverse the number* private static int reverseNumber(int N) {  
 int reverse = 0, reminder = 0;  
 boolean isNegative = N < 0;  
 if (isNegative) N = N \* (-1);  
 while (N > 0) {  
 reminder = N % 10;  
 reverse = reverse \* 10 + reminder;  
 N /= 10;  
 }  
 if (isNegative) reverse = reverse \* (-1);  
 return reverse;  
 }  
}

# Q2. <https://practice.geeksforgeeks.org/problems/pascal-triangle0652/1>

## Solution*:*

import java.util.ArrayList;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 16/05/2023  
 \*/  
/\*Given a positive integer N, return the Nth row of pascal's triangle.  
Pascal's triangle is a triangular array of the binomial coefficients formed by summing up the elements of previous row.  
  
Example :  
1  
1 1  
1 2 1  
1 3 3 1  
For N = 3, return 3rd row i.e 1 2 1  
\* \*/*public class QuestionTwoSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*nThRowOfPascalTriangle*(17));  
 }  
 *//to print whole pascal triangle*  
 private static void pascalTriangle(int N) {  
 for (int i = 0; i <= N; i++) {  
 for (int j = 0; j <= N - i; j++) {  
 System.*out*.print(" ");  
 }  
 for (int j = 0; j <= i; j++) {  
 System.*out*.print(" " + *factorial*(i) / (*factorial*(i - j) \* *factorial*(j)));  
 }  
 System.*out*.println();  
 }  
 }  
  
 private static long factorial(long i) {  
 if (i == 0) return 1;  
 return i \* *factorial*(i - 1);  
 }

*//to get the nth row of pascal triangle*  
 private static ArrayList<Long> nThRowOfPascalTriangle(int N) {  
 ArrayList<Long> arrayList = new ArrayList<>();  
 for (int i = 0; i <= N; i++) {  
 for (int j = 0; j <= i; j++) {  
 if (i == N - 1)  
 arrayList.add((*factorial*(i) / (*factorial*(i - j) \* *factorial*(j))));  
 }  
 }  
 return arrayList;  
 }  
}

# Q3. <https://leetcode.com/problems/richest-customer-wealth>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 16/05/2023  
 \*/*public class QuestionThreeSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*richestWealth*(new int[][]{{1, 2, 3}, {3, 2, 1}}));  
 System.*out*.println(*richestWealth*(new int[][]{{1, 5}, {7, 3}, {3, 5}}));  
 System.*out*.println(*richestWealth*(new int[][]{{2, 8, 7}, {7, 1, 3}, {1, 9, 5}}));  
 }  
  
 private static int richestWealth(int[][] accounts) {  
 int richest = 0;  
 for (int i = 0; i < accounts.length; i++) {  
 int sum = 0;  
 for (int j = 0; j < accounts[i].length; j++) {  
 sum += accounts[i][j];  
 }  
 if (richest < sum) richest = sum;  
 }  
 return richest;  
 }  
}

# Q4. <https://leetcode.com/problems/running-sum-of-1d-array/>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 16/05/2023  
 \*/*public class QuestionFourSolution {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*runningSum*(new int[]{1, 2, 3, 4})));  
 System.*out*.println(Arrays.*toString*(*runningSum*(new int[]{1, 1, 1, 1, 1})));  
 System.*out*.println(Arrays.*toString*(*runningSum*(new int[]{3, 1, 2, 10, 1})));  
 }  
  
 private static int[] runningSum(int[] nums) {  
 for (int i = 1; i < nums.length; i++) {  
 nums[i] = nums[i - 1] + nums[i];  
 }  
 return nums;  
 }  
}

# Q5. <https://leetcode.com/problems/jewels-and-stones>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 16/05/2023  
 \*/*public class QuestionFiveSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*jewelsAndStonesMethod1*("aA", "aAABbBBBbb"));  
 System.*out*.println(*jewelsAndStonesMethod2*("aAb", "aAABbBBBbb"));  
 System.*out*.println(*jewelsAndStonesMethod2*("z", "ZZ"));  
 }  
  
 static int jewelsAndStonesMethod1(String jewels, String stones) {  
 int count = 0;  
 for (char c1 : jewels.toCharArray()) {  
 for (char c2 : stones.toCharArray()) {  
 if (c1 == c2) count++;  
 }  
 }  
 return count;  
 }  
  
 static int jewelsAndStonesMethod2(String jewels, String stones) {  
 int n = stones.length();  
 int count = 0;  
 for (int i = 0; i < n; i++) {  
 if (jewels.indexOf(stones.charAt(i)) > -1) count++;  
 }  
 return count;  
 }  
}

# Q6. <https://leetcode.com/problems/minimum-absolute-difference>

## Solution:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 16/05/2023  
 \*/*public class QuestionSixSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*minimumAbsDiff*(new int[]{4, 2, 3, 1}));  
 System.*out*.println(*minimumAbsDiff*(new int[]{1, 3, 6, 10, 15}));  
 System.*out*.println(*minimumAbsDiff*(new int[]{3, 8, -10, 23, 19, -4, -14, 27}));  
 }  
  
 static List<List<Integer>> minimumAbsDiff(int[] arr) {  
 Arrays.*sort*(arr);a  
 int diff = Integer.*MAX\_VALUE*;  
 for (int i = 0; i < arr.length - 1; i++) {  
 diff = Math.*min*(diff, arr[i + 1] - arr[i]);  
 }  
 List<List<Integer>> result = new ArrayList<>();  
 for (int i = 0; i < arr.length - 1; i++) {  
 if (arr[i + 1] - arr[i] == diff) {  
 result.add(Arrays.*asList*(arr[i], arr[i + 1]));  
 }  
 }  
 return result;  
 }  
}

# Q7. <https://leetcode.com/problems/three-consecutive-odds>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 16/05/2023  
 \*/*public class QuestionSevenSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*threeConsecutiveOdd*(new int[]{2, 6, 4, 1}));  
 System.*out*.println(*threeConsecutiveOdd*(new int[]{1, 2, 34, 3, 4, 5, 7, 23, 12}));  
 System.*out*.println(*threeConsecutiveOdd*(new int[]{1, 2, 1, 1}));  
 }  
  
 static boolean threeConsecutiveOdd(int[] arr) {  
 int count = 0;  
 for (int i : arr) {  
 if (i % 2 != 0) {  
 count++;  
 if (count == 3) {  
 return true;  
 }  
 continue;  
 }  
 count = 0;  
 }  
 return false;  
 }  
}

# Q8. <https://leetcode.com/problems/move-zeroes>

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 16/05/2023  
 \*/*public class QuestionEightSolution {  
 public static void main(String[] args) {  
 System.*out*.println(Arrays.*toString*(*moveZeros*(new int[]{0, 1, 0, 3, 12})));  
 System.*out*.println(Arrays.*toString*(*moveZeros*(new int[]{0, -18, -22, 3, 12, 0, 0, 2, 9, 15})));  
 }  
  
 static int[] moveZeros(int[] nums) {  
 int count = 0;  
 for (int i = 0; i < nums.length; i++) {  
 if (nums[i] != 0) {  
 nums[count++] = nums[i];  
 }  
 }  
 while (count < nums.length) {  
 nums[count++] = 0;  
 }  
 return nums;  
 }  
}