|  |
| --- |
| Logical Questions with Code Code, Explanation, and Output |
| 1. Find the largest number in an array: |
| code |
|  |
| int findLargest(int[] arr) { |
| int largest = arr[0]; |
| for (int i = 1; i < arr.length; i++) { |
| if (arr[i] > largest) { |
| largest = arr[i]; |
| } |
| } |
| return largest; |
| } |
|  |
| // Example |
| int[] arr = {5, 2, 9, 1, 7}; |
| System.out.println(findLargest(arr)); // Output: 9 |
| 1. Check if an array contains a specific element: |
| code |
|  |
| boolean containsElement(int[] arr, int element) { |
| for (int num : arr) { |
| if (num == element) { |
| return true; |
| } |
| } |
| return false; |
| } |
|  |
| // Example |
| int[] arr = {1, 3, 5, 7}; |
| int element = 5; |
| System.out.println(containsElement(arr, element)); // Output: true |
| 1. Count the number of even numbers in an array: |
| code |
|  |
| int countEvenNumbers(int[] arr) { |
| int count = 0; |
| for (int num : arr) { |
| if (num % 2 == 0) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| int[] arr = {2, 3, 4, 5, 6}; |
| System.out.println(countEvenNumbers(arr)); // Output: 3 |
| 1. Reverse an array: |
| code |
|  |
| int[] reverseArray(int[] arr) { |
| int[] reversed = new int[arr.length]; |
| for (int i = 0; i < arr.length; i++) { |
| reversed[i] = arr[arr.length - 1 - i]; |
| } |
| return reversed; |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 3, 4}; |
| int[] reversed = reverseArray(arr); |
| for (int num : reversed) { |
| System.out.print(num + " "); // Output: 4 3 2 1 |
| } |
| 1. Find the sum of all elements in an array: |
| code |
|  |
| int sumArray(int[] arr) { |
| int sum = 0; |
| for (int num : arr) { |
| sum += num; |
| } |
| return sum; |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 3, 4}; |
| System.out.println(sumArray(arr)); // Output: 10 |
| 1. Find the average of elements in an array: |
| code |
|  |
| double averageArray(int[] arr) { |
| int sum = 0; |
| for (int num : arr) { |
| sum += num; |
| } |
| return (double) sum / arr.length; |
| } |
|  |
| // Example |
| int[] arr = {4, 5, 6}; |
| System.out.println(averageArray(arr)); // Output: 5.0 |
| 1. Check if a string is a palindrome (ignoring non-alphanumeric characters): |
| code |
|  |
| boolean isPalindrome(String str) { |
| String cleanedStr = str.replaceAll("[^a-zA-Z0-9]", "").toLowerCase(); |
| int left = 0; |
| int right = cleanedStr.length() - 1; |
| while (left < right) { |
| if (cleanedStr.charAt(left) != cleanedStr.charAt(right)) { |
| return false; |
| } |
| left++; |
| right--; |
| } |
| return true; |
| } |
|  |
| // Example |
| String str = "A man, a plan, a canal, Panama"; |
| System.out.println(isPalindrome(str)); // Output: true |
| 1. Count occurrences of a character in a string: |
| code |
|  |
| int countCharOccurrences(String str, char ch) { |
| int count = 0; |
| for (char c : str.toCharArray()) { |
| if (c == ch) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| String str = "hello world"; |
| char ch = 'l'; |
| System.out.println(countCharOccurrences(str, ch)); // Output: 3 |
| 1. Check if a year is a leap year: |
| code |
|  |
| boolean isLeapYear(int year) { |
| if (year % 4 == 0) { |
| if (year % 100 == 0) { |
| return year % 400 == 0; |
| } |
| return true; |
| } |
| return false; |
| } |
|  |
| // Example |
| int year = 2024; |
| System.out.println(isLeapYear(year)); // Output: true |
| 1. Find the day of the week for a given date: |
| code |
|  |
| String findDayOfWeek(int day, int month, int year) { |
| code.util.Calendar calendar = code.util.Calendar.getInstance(); |
| calendar.set(year, month - 1, day); |
| int dayOfWeek = calendar.get(code.util.Calendar.DAY\_OF\_WEEK); |
| switch (dayOfWeek) { |
| case code.util.Calendar.SUNDAY: |
| return "Sunday"; |
| case code.util.Calendar.MONDAY: |
| return "Monday"; |
| case code.util.Calendar.TUESDAY: |
| return "Tuesday"; |
| case code.util.Calendar.WEDNESDAY: |
| return "Wednesday"; |
| case code.util.Calendar.THURSDAY: |
| return "Thursday"; |
| case code.util.Calendar.FRIDAY: |
| return "Friday"; |
| case code.util.Calendar.SATURDAY: |
| return "Saturday"; |
| } |
| return "Unknown"; |
| } |
|  |
| // Example |
| int day = 25; |
| int month = 8; |
| int year = 2024; |
| System.out.println(findDayOfWeek(day, month, year)); // Output: Sunday |
| 1. Find the factorial of a number: |
| code |
|  |
| int factorial(int num) { |
| int result = 1; |
| for (int i = 1; i <= num; i++) { |
| result \*= i; |
| } |
| return result; |
| } |
|  |
| // Example |
| int num = 5; |
| System.out.println(factorial(num)); // Output: 120 |
| 1. Check if an array is sorted in ascending order: |
| code |
|  |
| boolean isSorted(int[] arr) { |
| for (int i = 1; i < arr.length; i++) { |
| if (arr[i] < arr[i - 1]) { |
| return false; |
| } |
| } |
| return true; |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 3, 4, 5}; |
| System.out.println(isSorted(arr)); // Output: true |
| 1. Find the minimum and maximum in an array: |
| code |
|  |
| int[] findMinMax(int[] arr) { |
| int min = arr[0]; |
| int max = arr[0]; |
| for (int num : arr) { |
| if (num < min) min = num; |
| if (num > max) max = num; |
| } |
| return new int[]{min, max}; |
| } |
|  |
| // Example |
| int[] arr = {2, 8, 1, 7}; |
| int[] minMax = findMinMax(arr); |
| System.out.println("Min: " + minMax[0] + ", Max: " + minMax[1]); // Output: Min: 1, Max: 8 |
| 1. Count the number of vowels in a string: |
| code |
|  |
| int countVowels(String str) { |
| int count = 0; |
| for (char c : str.toLowerCase().toCharArray()) { |
| if ("aeiou".indexOf(c) != -1) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| String str = "Hello World"; |
| System.out.println(countVowels(str)); // Output: 3 |
| 1. Find the common elements between two arrays: |
| code |
|  |
| int[] findCommonElements(int[] arr1, int[] arr2) { |
| int[] common = new int[Math.min(arr1.length, arr2.length)]; |
| int index = 0; |
| for (int num1 : arr1) { |
| for (int num2 : arr2) { |
| if (num1 == num2) { |
| common[index++] = num1; |
| } |
| } |
| } |
| return code.util.Arrays.copyOf(common, index); |
| } |
|  |
| // Example |
| int[] arr1 = {1, 2, 3, 4}; |
| int[] arr2 = {3, 4, 5, 6}; |
| int[] common = findCommonElements(arr1, arr2); |
| for (int num : common) { |
| System.out.print(num + " "); // Output: 3 4 |
| } |
| 1. Check if a string contains only uppercase letters: |
| code |
|  |
| boolean isAllUpperCase(String str) { |
| for (char c : str.toCharArray()) { |
| if (!Character.isUpperCase(c)) { |
| return false; |
| } |
| } |
| return true; |
| } |
|  |
| // Example |
| String str = "HELLO"; |
| System.out.println(isAllUpperCase(str)); // Output: true |
| 1. Count the number of words in a string: |
| code |
|  |
| int countWords(String str) { |
| String[] words = str.trim().split("\\s+"); |
| return words.length; |
| } |
|  |
| // Example |
| String str = "Code is a programming language"; |
| System.out.println(countWords(str)); // Output: 5 |
| 1. Check if a string contains only whitespace characters: |
| code |
|  |
| boolean isAllWhitespace(String str) { |
| for (char c : str.toCharArray()) { |
| if (!Character.isWhitespace(c)) { |
| return false; |
| } |
| } |
| return true; |
| } |
|  |
| // Example |
| String str = " "; |
| System.out.println(isAllWhitespace(str)); // Output: true |
| 1. Find the longest word in a string: |
| code |
|  |
| String findLongestWord(String str) { |
| String[] words = str.trim().split("\\s+"); |
| String longest = ""; |
| for (String word : words) { |
| if (word.length() > longest.length()) { |
| longest = word; |
| } |
| } |
| return longest; |
| } |
|  |
| // Example |
| String str = "Find the longest word in this sentence"; |
| System.out.println(findLongestWord(str)); // Output: sentence |
| 1. Calculate the difference in days between two dates: |
| code |
|  |
| int calculateDateDifference(code.util.Date date1, code.util.Date date2) { |
| long differenceInMillis = Math.abs(date1.getTime() - date2.getTime()); |
| return (int) (differenceInMillis / (1000 \* 60 \* 60 \* 24)); |
| } |
|  |
| // Example |
| code.util.Calendar cal1 = code.util.Calendar.getInstance(); |
| code.util.Calendar cal2 = code.util.Calendar.getInstance(); |
| cal1.set(2024, 7, 25); // August 25, 2024 |
| cal2.set(2024, 8, 25); // September 25, 2024 |
| System.out.println(calculateDateDifference(cal1.getTime(), cal2.getTime())); // Output: 31 |
|  |
|  |
| 21. **Check if a string contains only digits:** |
| code |
|  |
| boolean isAllDigits(String str) { |
| for (char c : str.toCharArray()) { |
| if (!Character.isDigit(c)) { |
| return false; |
| } |
| } |
| return true; |
| } |
|  |
| // Example |
| String str = "123456"; |
| System.out.println(isAllDigits(str)); // Output: true |
| 22. **Find the factorial of a number using recursion:** |
| code |
|  |
| int factorial(int num) { |
| if (num == 0) { |
| return 1; |
| } |
| return num \* factorial(num - 1); |
| } |
|  |
| // Example |
| int num = 4; |
| System.out.println(factorial(num)); // Output: 24 |
| 23. **Check if two strings are anagrams of each other:** |
| code |
|  |
| boolean areAnagrams(String str1, String str2) { |
| char[] arr1 = str1.replaceAll("[^a-zA-Z]", "").toLowerCase().toCharArray(); |
| char[] arr2 = str2.replaceAll("[^a-zA-Z]", "").toLowerCase().toCharArray(); |
| code.util.Arrays.sort(arr1); |
| code.util.Arrays.sort(arr2); |
| return code.util.Arrays.equals(arr1, arr2); |
| } |
|  |
| // Example |
| String str1 = "Listen"; |
| String str2 = "Silent"; |
| System.out.println(areAnagrams(str1, str2)); // Output: true |
| 24. **Count the number of digits in a string:** |
| code |
|  |
| int countDigits(String str) { |
| int count = 0; |
| for (char c : str.toCharArray()) { |
| if (Character.isDigit(c)) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| String str = "a1b2c3"; |
| System.out.println(countDigits(str)); // Output: 3 |
| 25. **Find the largest prime number less than a given number:** |
| code |
|  |
| boolean isPrime(int num) { |
| if (num <= 1) return false; |
| for (int i = 2; i <= Math.sqrt(num); i++) { |
| if (num % i == 0) return false; |
| } |
| return true; |
| } |
|  |
| int largestPrimeLessThan(int num) { |
| for (int i = num - 1; i > 1; i--) { |
| if (isPrime(i)) return i; |
| } |
| return -1; |
| } |
|  |
| // Example |
| int num = 20; |
| System.out.println(largestPrimeLessThan(num)); // Output: 19 |
| 26. **Convert a string to a character array and reverse it:** |
| code |
|  |
| char[] reverseStringToCharArray(String str) { |
| char[] arr = str.toCharArray(); |
| int left = 0, right = arr.length - 1; |
| while (left < right) { |
| char temp = arr[left]; |
| arr[left] = arr[right]; |
| arr[right] = temp; |
| left++; |
| right--; |
| } |
| return arr; |
| } |
|  |
| // Example |
| String str = "hello"; |
| char[] reversed = reverseStringToCharArray(str); |
| System.out.println(new String(reversed)); // Output: olleh |
| 27. **Calculate the power of a number using a loop:** |
| code |
|  |
| int power(int base, int exponent) { |
| int result = 1; |
| for (int i = 0; i < exponent; i++) { |
| result \*= base; |
| } |
| return result; |
| } |
|  |
| // Example |
| int base = 2; |
| int exponent = 3; |
| System.out.println(power(base, exponent)); // Output: 8 |
| 28. **Check if a string is a valid email format:** |
| code |
|  |
| boolean isValidEmail(String str) { |
| if (str == null || !str.contains("@") || !str.contains(".")) { |
| return false; |
| } |
| String[] parts = str.split("@"); |
| return parts.length == 2 && parts[1].contains("."); |
| } |
|  |
| // Example |
| String email = "example@example.com"; |
| System.out.println(isValidEmail(email)); // Output: true |
| 29. **Find the number of days in a given month of a given year:** |
| code |
|  |
| int daysInMonth(int month, int year) { |
| int[] daysInMonth = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}; |
| if (month == 2 && isLeapYear(year)) { |
| return 29; |
| } |
| return daysInMonth[month - 1]; |
| } |
|  |
| boolean isLeapYear(int year) { |
| return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0); |
| } |
|  |
| // Example |
| int month = 2; |
| int year = 2024; |
| System.out.println(daysInMonth(month, year)); // Output: 29 |
| 30. **Convert a given number of seconds into hours, minutes, and seconds:** |
| code |
|  |
| void convertSeconds(int totalSeconds) { |
| int hours = totalSeconds / 3600; |
| int minutes = (totalSeconds % 3600) / 60; |
| int seconds = totalSeconds % 60; |
| System.out.println(hours + " hours, " + minutes + " minutes, " + seconds + " seconds"); |
| } |
|  |
| // Example |
| int totalSeconds = 3665; |
| convertSeconds(totalSeconds); // Output: 1 hours, 1 minutes, 5 seconds |
| 31. **Find the common elements between two arrays and return their count:** |
| code |
|  |
| int countCommonElements(int[] arr1, int[] arr2) { |
| int count = 0; |
| for (int num1 : arr1) { |
| for (int num2 : arr2) { |
| if (num1 == num2) { |
| count++; |
| break; |
| } |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| int[] arr1 = {1, 2, 3}; |
| int[] arr2 = {2, 3, 4}; |
| System.out.println(countCommonElements(arr1, arr2)); // Output: 2 |
| 32. **Check if a string contains only alphabetic characters:** |
| code |
|  |
| boolean isAllAlphabetic(String str) { |
| for (char c : str.toCharArray()) { |
| if (!Character.isLetter(c)) { |
| return false; |
| } |
| } |
| return true; |
| } |
|  |
| // Example |
| String str = "Hello"; |
| System.out.println(isAllAlphabetic(str)); // Output: true |
| 33. **Find the number of prime numbers in an array:** |
| code |
|  |
| int countPrimes(int[] arr) { |
| int count = 0; |
| for (int num : arr) { |
| if (isPrime(num)) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| int[] arr = {2, 3, 4, 5, 6}; |
| System.out.println(countPrimes(arr)); // Output: 3 |
| 34. **Find the sum of digits of a number:** |
| code |
|  |
| int sumOfDigits(int num) { |
| int sum = 0; |
| while (num > 0) { |
| sum += num % 10; |
| num /= 10; |
| } |
| return sum; |
| } |
|  |
| // Example |
| int num = 1234; |
| System.out.println(sumOfDigits(num)); // Output: 10 |
| 35. **Convert a string to uppercase and lowercase:** |
| code |
|  |
| void convertCase(String str) { |
| System.out.println("Uppercase: " + str.toUpperCase()); |
| System.out.println("Lowercase: " + str.toLowerCase()); |
| } |
|  |
| // Example |
| String str = "Code"; |
| convertCase(str); |
| // Output: |
| // Uppercase: CODE |
| // Lowercase: code |
| 36. **Check if a number is a perfect square:** |
| code |
|  |
| boolean isPerfectSquare(int num) { |
| int sqrt = (int) Math.sqrt(num); |
| return num == sqrt \* sqrt; |
| } |
|  |
| // Example |
| int num = 16; |
| System.out.println(isPerfectSquare(num)); // Output: true |
| 37. **Find the most frequent character in a string:** |
| code |
|  |
| char mostFrequentChar(String str) { |
| int[] frequency = new int[256]; |
| for (char c : str.toCharArray()) { |
| frequency[c]++; |
| } |
| int maxFreq = 0; |
| char result = ' '; |
| for (int i = 0; i < frequency.length; i++) { |
| if (frequency[i] > maxFreq) { |
| maxFreq = frequency[i]; |
| result = (char) i; |
| } |
| } |
| return result; |
| } |
|  |
| // Example |
| String str = "abracadabra"; |
| System.out.println(mostFrequentChar(str)); // Output: a |
| 38. **Find the date one week from today:** |
| code |
|  |
| code.util.Date oneWeekFromToday() { |
| code.util.Calendar cal = code.util.Calendar.getInstance(); |
| cal.add(code.util.Calendar.WEEK\_OF\_YEAR, 1); |
| return cal.getTime(); |
| } |
|  |
| // Example |
| System.out.println(oneWeekFromToday()); // Output: (Date one week from today) |
| 39. **Find the difference in hours between two dates:** |
| code |
|  |
| int calculateHoursDifference(code.util.Date date1, code.util.Date date2) { |
| long differenceInMillis = Math.abs(date1.getTime() - date2.getTime()); |
| return (int) (differenceInMillis / (1000 \* 60 \* 60)); |
| } |
|  |
| // Example |
| code.util.Calendar cal1 = code.util.Calendar.getInstance(); |
| code.util.Calendar cal2 = code.util.Calendar.getInstance(); |
| cal1.set(2024, 7, 25, 8, 0); // August 25, 2024, 08:00 |
| cal2.set(2024, 7, 25, 12, 0); // August 25, 2024, 12:00 |
| System.out.println(calculateHoursDifference(cal1.getTime(), cal2.getTime())); // Output: 4 |
| 40. **Check if a year is a leap year:** |
| code |
|  |
| boolean isLeapYear(int year) { |
| return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0); |
| } |
|  |
| // Example |
| int year = 2024; |
| System.out.println(isLeapYear(year)); // Output: true |
| 41. **Count the number of vowels in a string:** |
| code |
|  |
| int countVowels(String str) { |
| int count = 0; |
| String vowels = "aeiouAEIOU"; |
| for (char c : str.toCharArray()) { |
| if (vowels.indexOf(c) != -1) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| String str = "Hello World"; |
| System.out.println(countVowels(str)); // Output: 3 |
| 42. **Find the smallest number in an array:** |
| code |
|  |
| int findSmallest(int[] arr) { |
| int min = arr[0]; |
| for (int num : arr) { |
| if (num < min) { |
| min = num; |
| } |
| } |
| return min; |
| } |
|  |
| // Example |
| int[] arr = {3, 1, 4, 1, 5, 9}; |
| System.out.println(findSmallest(arr)); // Output: 1 |
| 43. **Check if a string starts with a specific prefix:** |
| code |
|  |
| boolean startsWith(String str, String prefix) { |
| return str.startsWith(prefix); |
| } |
|  |
| // Example |
| String str = "Hello, World!"; |
| String prefix = "Hello"; |
| System.out.println(startsWith(str, prefix)); // Output: true |
| 44. **Calculate the sum of even numbers in an array:** |
| code |
|  |
| int sumOfEvens(int[] arr) { |
| int sum = 0; |
| for (int num : arr) { |
| if (num % 2 == 0) { |
| sum += num; |
| } |
| } |
| return sum; |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 3, 4, 5}; |
| System.out.println(sumOfEvens(arr)); // Output: 6 |
| 45. **Check if a string is a valid palindrome:** |
| code |
|  |
| boolean isPalindrome(String str) { |
| String cleaned = str.replaceAll("[^a-zA-Z0-9]", "").toLowerCase(); |
| String reversed = new StringBuilder(cleaned).reverse().toString(); |
| return cleaned.equals(reversed); |
| } |
|  |
| // Example |
| String str = "A man, a plan, a canal, Panama"; |
| System.out.println(isPalindrome(str)); // Output: true |
| 46. **Find the number of days between two dates:** |
| code |
|  |
| int daysBetween(code.util.Date date1, code.util.Date date2) { |
| long differenceInMillis = Math.abs(date1.getTime() - date2.getTime()); |
| return (int) (differenceInMillis / (1000 \* 60 \* 60 \* 24)); |
| } |
|  |
| // Example |
| code.util.Calendar cal1 = code.util.Calendar.getInstance(); |
| code.util.Calendar cal2 = code.util.Calendar.getInstance(); |
| cal1.set(2024, 7, 1); // August 1, 2024 |
| cal2.set(2024, 7, 25); // August 25, 2024 |
| System.out.println(daysBetween(cal1.getTime(), cal2.getTime())); // Output: 24 |
| 47. **Check if a number is a power of two:** |
| code |
|  |
| boolean isPowerOfTwo(int num) { |
| return num > 0 && (num & (num - 1)) == 0; |
| } |
|  |
| // Example |
| int num = 16; |
| System.out.println(isPowerOfTwo(num)); // Output: true |
| 48. **Find the median of an array of numbers:** |
| code |
|  |
| double findMedian(int[] arr) { |
| code.util.Arrays.sort(arr); |
| int middle = arr.length / 2; |
| if (arr.length % 2 == 0) { |
| return (arr[middle - 1] + arr[middle]) / 2.0; |
| } else { |
| return arr[middle]; |
| } |
| } |
|  |
| // Example |
| int[] arr = {3, 1, 4, 1, 5}; |
| System.out.println(findMedian(arr)); // Output: 3 |
| 49. **Check if a string contains a specific substring:** |
| code |
|  |
| boolean containsSubstring(String str, String substring) { |
| return str.contains(substring); |
| } |
|  |
| // Example |
| String str = "Hello, World!"; |
| String substring = "World"; |
| System.out.println(containsSubstring(str, substring)); // Output: true |
| 50. **Calculate the area of a circle given its radius:** |
| code |
|  |
| double calculateAreaOfCircle(double radius) { |
| return Math.PI \* radius \* radius; |
| } |
|  |
| // Example |
| double radius = 5.0; |
| System.out.println(calculateAreaOfCircle(radius)); // Output: 78.53981633974483 |
| 51. **Reverse an integer number:** |
| code |
|  |
| int reverseNumber(int num) { |
| int reversed = 0; |
| while (num != 0) { |
| reversed = reversed \* 10 + num % 10; |
| num /= 10; |
| } |
| return reversed; |
| } |
|  |
| // Example |
| int num = 1234; |
| System.out.println(reverseNumber(num)); // Output: 4321 |
| 52. **Check if a number is divisible by another number:** |
| code |
|  |
| boolean isDivisible(int num1, int num2) { |
| return num1 % num2 == 0; |
| } |
|  |
| // Example |
| int num1 = 10; |
| int num2 = 2; |
| System.out.println(isDivisible(num1, num2)); // Output: true |
| 53. **Find the sum of the squares of numbers in an array:** |
| code |
|  |
| int sumOfSquares(int[] arr) { |
| int sum = 0; |
| for (int num : arr) { |
| sum += num \* num; |
| } |
| return sum; |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 3}; |
| System.out.println(sumOfSquares(arr)); // Output: 14 |
| 54. **Check if a string contains only uppercase letters:** |
| code |
|  |
| boolean isUppercase(String str) { |
| return str.equals(str.toUpperCase()); |
| } |
|  |
| // Example |
| String str = "HELLO"; |
| System.out.println(isUppercase(str)); // Output: true |
| 55. **Find the greatest common divisor (GCD) of two numbers:** |
| code |
|  |
| int gcd(int a, int b) { |
| while (b != 0) { |
| int temp = b; |
| b = a % b; |
| a = temp; |
| } |
| return a; |
| } |
|  |
| // Example |
| int a = 48; |
| int b = 18; |
| System.out.println(gcd(a, b)); // Output: 6 |
| 56. **Count the number of uppercase letters in a string:** |
| code |
|  |
| int countUppercase(String str) { |
| int count = 0; |
| for (char c : str.toCharArray()) { |
| if (Character.isUpperCase(c)) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| String str = "Hello World"; |
| System.out.println(countUppercase(str)); // Output: 2 |
| 57. **Find the intersection of two integer arrays:** |
| code |
|  |
| int[] intersectArrays(int[] arr1, int[] arr2) { |
| int[] temp = new int[Math.min(arr1.length, arr2.length)]; |
| int count = 0; |
| for (int num1 : arr1) { |
| for (int num2 : arr2) { |
| if (num1 == num2) { |
| temp[count++] = num1; |
| break; |
| } |
| } |
| } |
| int[] result = new int[count]; |
| System.arraycopy(temp, 0, result, 0, count); |
| return result; |
| } |
|  |
| // Example |
| int[] arr1 = {1, 2, 3}; |
| int[] arr2 = {2, 3, 4}; |
| int[] intersection = intersectArrays(arr1, arr2); |
| System.out.println(code.util.Arrays.toString(intersection)); // Output: [2, 3] |
| 58. **Calculate the sum of odd numbers in an array:** |
| code |
|  |
| int sumOfOdds(int[] arr) { |
| int sum = 0; |
| for (int num : arr) { |
| if (num % 2 != 0) { |
| sum += num; |
| } |
| } |
| return sum; |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 3, 4, 5}; |
| System.out.println(sumOfOdds(arr)); // Output: 9 |
| 59. Count the number of words in a string: |
| code |
|  |
| int countWords(String str) { |
| String[] words = str.trim().split("\\s+"); |
| return words.length; |
| } |
|  |
| // Example |
| String str = "Hello world, how are you?"; |
| System.out.println(countWords(str)); // Output: 5 |
| 60. Calculate the average of numbers in an array: |
| code |
|  |
| double calculateAverage(int[] arr) { |
| int sum = 0; |
| for (int num : arr) { |
| sum += num; |
| } |
| return (double) sum / arr.length; |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 3, 4, 5}; |
| System.out.println(calculateAverage(arr)); // Output: 3.0 |
| 61. Check if a string contains only lowercase letters: |
| code |
|  |
| boolean isLowercase(String str) { |
| return str.equals(str.toLowerCase()); |
| } |
|  |
| // Example |
| String str = "hello"; |
| System.out.println(isLowercase(str)); // Output: true |
| 62. Find the median of an array of numbers: |
| code |
|  |
| double findMedian(int[] arr) { |
| code.util.Arrays.sort(arr); |
| int middle = arr.length / 2; |
| if (arr.length % 2 == 0) { |
| return (arr[middle - 1] + arr[middle]) / 2.0; |
| } else { |
| return arr[middle]; |
| } |
| } |
|  |
| // Example |
| int[] arr = {3, 1, 4, 1, 5}; |
| System.out.println(findMedian(arr)); // Output: 3 |
| 63. Calculate the sum of digits of a number: |
| code |
|  |
| int sumOfDigits(int num) { |
| int sum = 0; |
| while (num > 0) { |
| sum += num % 10; |
| num /= 10; |
| } |
| return sum; |
| } |
|  |
| // Example |
| int num = 123; |
| System.out.println(sumOfDigits(num)); // Output: 6 |
| 64. Check if a string is a valid IPv4 address: |
| code |
|  |
| boolean isValidIPv4(String str) { |
| String[] parts = str.split("\\."); |
| if (parts.length != 4) return false; |
| for (String part : parts) { |
| try { |
| int num = Integer.parseInt(part); |
| if (num < 0 || num > 255) return false; |
| } catch (NumberFormatException e) { |
| return false; |
| } |
| } |
| return true; |
| } |
|  |
| // Example |
| String ip = "192.168.1.1"; |
| System.out.println(isValidIPv4(ip)); // Output: true |
| 65. Calculate the number of days in a given month and year: |
| code |
|  |
| int daysInMonth(int month, int year) { |
| int[] daysInMonth = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}; |
| if (month == 2 && isLeapYear(year)) { |
| return 29; |
| } |
| return daysInMonth[month - 1]; |
| } |
|  |
| boolean isLeapYear(int year) { |
| return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0); |
| } |
|  |
| // Example |
| int month = 2; |
| int year = 2024; |
| System.out.println(daysInMonth(month, year)); // Output: 29 |
| 66. Convert a string to title case (capitalize the first letter of each word): |
| code |
|  |
| String toTitleCase(String str) { |
| String[] words = str.split(" "); |
| StringBuilder titleCase = new StringBuilder(); |
| for (String word : words) { |
| if (!word.isEmpty()) { |
| titleCase.append(Character.toUpperCase(word.charAt(0))) |
| .append(word.substring(1).toLowerCase()) |
| .append(" "); |
| } |
| } |
| return titleCase.toString().trim(); |
| } |
|  |
| // Example |
| String str = "hello world"; |
| System.out.println(toTitleCase(str)); // Output: Hello World |
| 67. Find the sum of elements in a 2D array: |
| code |
|  |
| int sum2DArray(int[][] arr) { |
| int sum = 0; |
| for (int[] row : arr) { |
| for (int num : row) { |
| sum += num; |
| } |
| } |
| return sum; |
| } |
|  |
| // Example |
| int[][] arr = {{1, 2, 3}, {4, 5, 6}}; |
| System.out.println(sum2DArray(arr)); // Output: 21 |
| 68. Count the number of characters in a string excluding spaces: |
| code |
|  |
| int countNonSpaceChars(String str) { |
| int count = 0; |
| for (char c : str.toCharArray()) { |
| if (c != ' ') { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| String str = "Hello World"; |
| System.out.println(countNonSpaceChars(str)); // Output: 10 |
| 69. Check if a string is a valid phone number (basic validation): |
| code |
|  |
| boolean isValidPhoneNumber(String str) { |
| return str.matches("\\d{10}"); |
| } |
|  |
| // Example |
| String phoneNumber = "1234567890"; |
| System.out.println(isValidPhoneNumber(phoneNumber)); // Output: true |
| 70. Find the largest number in a 2D array: |
| code |
|  |
| int findLargest2DArray(int[][] arr) { |
| int max = arr[0][0]; |
| for (int[] row : arr) { |
| for (int num : row) { |
| if (num > max) { |
| max = num; |
| } |
| } |
| } |
| return max; |
| } |
|  |
| // Example |
| int[][] arr = {{1, 2, 3}, {4, 5, 6}}; |
| System.out.println(findLargest2DArray(arr)); // Output: 6 |
| 71. Check if a string contains only numeric characters: |
| code |
|  |
| boolean isNumeric(String str) { |
| return str.chars().allMatch(Character::isDigit); |
| } |
|  |
| // Example |
| String str = "123456"; |
| System.out.println(isNumeric(str)); // Output: true |
| 72. Find the smallest number in an array: |
| code |
|  |
| int findSmallest(int[] arr) { |
| int min = arr[0]; |
| for (int num : arr) { |
| if (num < min) { |
| min = num; |
| } |
| } |
| return min; |
| } |
|  |
| // Example |
| int[] arr = {5, 2, 9, 1}; |
| System.out.println(findSmallest(arr)); // Output: 1 |
| 73. Find the largest number in an array: |
| code |
|  |
| int findLargest(int[] arr) { |
| int max = arr[0]; |
| for (int num : arr) { |
| if (num > max) { |
| max = num; |
| } |
| } |
| return max; |
| } |
|  |
| // Example |
| int[] arr = {5, 2, 9, 1}; |
| System.out.println(findLargest(arr)); // Output: 9 |
| 74. Find the sum of all even numbers in a range: |
| code |
|  |
| int sumOfEvensInRange(int start, int end) { |
| int sum = 0; |
| for (int i = start; i <= end; i++) { |
| if (i % 2 == 0) { |
| sum += i; |
| } |
| } |
| return sum; |
| } |
|  |
| // Example |
| int start = 1; |
| int end = 10; |
| System.out.println(sumOfEvensInRange(start, end)); // Output: 30 |
| 75. Check if a string is a valid hexadecimal number: |
| code |
|  |
| boolean isHexadecimal(String str) { |
| return str.matches("[0-9A-Fa-f]+"); |
| } |
|  |
| // Example |
| String hex = "1A3F"; |
| System.out.println(isHexadecimal(hex)); // Output: true |
| 76. Find the number of characters in a string: |
| code |
|  |
| int countCharacters(String str) { |
| return str.length(); |
| } |
|  |
| // Example |
| String str = "Hello"; |
| System.out.println(countCharacters(str)); // Output: 5 |
| 77. Find the product of all numbers in an array: |
| code |
|  |
| int productOfArray(int[] arr) { |
| int product = 1; |
| for (int num : arr) { |
| product \*= num; |
| } |
| return product; |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 3, 4}; |
| System.out.println(productOfArray(arr)); // Output: 24 |
| 78. Check if a string is a valid date in YYYY-MM-DD format: |
| code |
|  |
| boolean isValidDate(String str) { |
| try { |
| code.time.LocalDate.parse(str); |
| return true; |
| } catch (code.time.format.DateTimeParseException e) { |
| return false; |
| } |
| } |
|  |
| // Example |
| String date = "2024-08-25"; |
| System.out.println(isValidDate(date)); // Output: true |
| 79. Calculate the area of a rectangle: |
| code |
|  |
| int calculateAreaOfRectangle(int width, int height) { |
| return width \* height; |
| } |
|  |
| // Example |
| int width = 5; |
| int height = 10; |
| System.out.println(calculateAreaOfRectangle(width, height)); // Output: 50 |
| 80. Find the largest prime number less than a given number: |
| code |
|  |
| int largestPrimeLessThan(int num) { |
| for (int i = num - 1; i > 1; i--) { |
| if (isPrime(i)) { |
| return i; |
| } |
| } |
| return -1; |
| } |
|  |
| boolean isPrime(int num) { |
| if (num <= 1) return false; |
| for (int i = 2; i <= Math.sqrt(num); i++) { |
| if (num % i == 0) return false; |
| } |
| return true; |
| } |
|  |
| // Example |
| int num = 20; |
| System.out.println(largestPrimeLessThan(num)); // Output: 19 |
| 81. Find the number of digits in an integer: |
| code |
|  |
| int countDigits(int num) { |
| return String.valueOf(num).length(); |
| } |
|  |
| // Example |
| int num = 12345; |
| System.out.println(countDigits(num)); // Output: 5 |
| 82. Find the sum of the first n natural numbers: |
| code |
|  |
| int sumOfFirstNNumbers(int n) { |
| return n \* (n + 1) / 2; |
| } |
|  |
| // Example |
| int n = 5; |
| System.out.println(sumOfFirstNNumbers(n)); // Output: 15 |
| 83. Check if a number is an Armstrong number: |
| code |
|  |
| boolean isArmstrong(int num) { |
| int original = num; |
| int sum = 0; |
| int digits = String.valueOf(num).length(); |
| while (num > 0) { |
| int digit = num % 10; |
| sum += Math.pow(digit, digits); |
| num /= 10; |
| } |
| return sum == original; |
| } |
|  |
| // Example |
| int num = 153; |
| System.out.println(isArmstrong(num)); // Output: true |
| 84. Find the common elements between two arrays: |
| code |
|  |
| int[] commonElements(int[] arr1, int[] arr2) { |
| code.util.Set<Integer> set1 = new code.util.HashSet<>(); |
| code.util.Set<Integer> common = new code.util.HashSet<>(); |
| for (int num : arr1) { |
| set1.add(num); |
| } |
| for (int num : arr2) { |
| if (set1.contains(num)) { |
| common.add(num); |
| } |
| } |
| return common.stream().mapToInt(Integer::intValue).toArray(); |
| } |
|  |
| // Example |
| int[] arr1 = {1, 2, 3, 4}; |
| int[] arr2 = {3, 4, 5, 6}; |
| int[] common = commonElements(arr1, arr2); |
| System.out.println(code.util.Arrays.toString(common)); // Output: [3, 4] |
| 85. Check if a number is a perfect number: |
| code |
|  |
| boolean isPerfectNumber(int num) { |
| int sum = 0; |
| for (int i = 1; i <= num / 2; i++) { |
| if (num % i == 0) { |
| sum += i; |
| } |
| } |
| return sum == num; |
| } |
|  |
| // Example |
| int num = 28; |
| System.out.println(isPerfectNumber(num)); // Output: true |
| 86. Find the largest element in a 2D array: |
| code |
|  |
| int findLargestIn2DArray(int[][] arr) { |
| int max = arr[0][0]; |
| for (int[] row : arr) { |
| for (int num : row) { |
| if (num > max) { |
| max = num; |
| } |
| } |
| } |
| return max; |
| } |
|  |
| // Example |
| int[][] arr = {{1, 2, 3}, {4, 5, 6}}; |
| System.out.println(findLargestIn2DArray(arr)); // Output: 6 |
| 87. Check if a number is a palindrome: |
| code |
|  |
| boolean isPalindrome(int num) { |
| int original = num; |
| int reversed = 0; |
| while (num != 0) { |
| reversed = reversed \* 10 + num % 10; |
| num /= 10; |
| } |
| return original == reversed; |
| } |
|  |
| // Example |
| int num = 121; |
| System.out.println(isPalindrome(num)); // Output: true |
| 88. Find the factorial of a number: |
| code |
|  |
| long factorial(int num) { |
| long result = 1; |
| for (int i = 1; i <= num; i++) { |
| result \*= i; |
| } |
| return result; |
| } |
|  |
| // Example |
| int num = 5; |
| System.out.println(factorial(num)); // Output: 120 |
| 89. Find the length of the longest string in an array: |
| code |
|  |
| int lengthOfLongestString(String[] arr) { |
| int maxLength = 0; |
| for (String str : arr) { |
| if (str.length() > maxLength) { |
| maxLength = str.length(); |
| } |
| } |
| return maxLength; |
| } |
|  |
| // Example |
| String[] arr = {"short", "longer", "longest"}; |
| System.out.println(lengthOfLongestString(arr)); // Output: 7 |
| 90. Count the number of unique elements in an array: |
| code |
|  |
| int countUnique(int[] arr) { |
| code.util.Set<Integer> uniqueElements = new code.util.HashSet<>(); |
| for (int num : arr) { |
| uniqueElements.add(num); |
| } |
| return uniqueElements.size(); |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 2, 3, 4, 4}; |
| System.out.println(countUnique(arr)); // Output: 4 |
| 91. Find the common elements between two lists: |
|  |
| code.util.List<Integer> commonElements(code.util.List<Integer> list1, code.util.List<Integer> list2) { |
| code.util.Set<Integer> set1 = new code.util.HashSet<>(list1); |
| code.util.Set<Integer> common = new code.util.HashSet<>(); |
| for (int num : list2) { |
| if (set1.contains(num)) { |
| common.add(num); |
| } |
| } |
| return new code.util.ArrayList<>(common); |
| } |
|  |
| // Example |
| code.util.List<Integer> list1 = code.util.Arrays.asList(1, 2, 3, 4); |
| code.util.List<Integer> list2 = code.util.Arrays.asList(3, 4, 5, 6); |
| System.out.println(commonElements(list1, list2)); // Output: [3, 4] |
| 92. Check if a number is a strong number (sum of the factorials of its digits equals the number itself): |
| code |
|  |
| boolean isStrongNumber(int num) { |
| int original = num; |
| int sum = 0; |
| while (num > 0) { |
| int digit = num % 10; |
| sum += factorial(digit); |
| num /= 10; |
| } |
| return sum == original; |
| } |
|  |
| long factorial(int num) { |
| long result = 1; |
| for (int i = 1; i <= num; i++) { |
| result \*= i; |
| } |
| return result; |
| } |
|  |
| // Example |
| int num = 145; |
| System.out.println(isStrongNumber(num)); // Output: true |
| 93. Find the most frequent element in an array: |
| code |
|  |
| int mostFrequentElement(int[] arr) { |
| code.util.Map<Integer, Integer> frequencyMap = new code.util.HashMap<>(); |
| for (int num : arr) { |
| frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1); |
| } |
| return frequencyMap.entrySet().stream() |
| .max(code.util.Map.Entry.comparingByValue()) |
| .map(code.util.Map.Entry::getKey) |
| .orElseThrow(); |
| } |
|  |
| // Example |
| int[] arr = {1, 2, 2, 3, 3, 3}; |
| System.out.println(mostFrequentElement(arr)); // Output: 3 |
| 94. Find the smallest prime number greater than a given number: |
| code |
|  |
| int smallestPrimeGreaterThan(int num) { |
| int candidate = num + 1; |
| while (!isPrime(candidate)) { |
| candidate++; |
| } |
| return candidate; |
| } |
|  |
| boolean isPrime(int num) { |
| if (num <= 1) return false; |
| for (int i = 2; i <= Math.sqrt(num); i++) { |
| if (num % i == 0) return false; |
| } |
| return true; |
| } |
|  |
| // Example |
| int num = 10; |
| System.out.println(smallestPrimeGreaterThan(num)); // Output: 11 |
| 95. Find the longest substring without repeating characters: |
| code |
|  |
| String longestUniqueSubstring(String str) { |
| int n = str.length(); |
| int start = 0, maxLength = 0; |
| String longest = ""; |
| code.util.Set<Character> seen = new code.util.HashSet<>(); |
| for (int end = 0; end < n; end++) { |
| while (seen.contains(str.charAt(end))) { |
| seen.remove(str.charAt(start)); |
| start++; |
| } |
| seen.add(str.charAt(end)); |
| if (end - start + 1 > maxLength) { |
| maxLength = end - start + 1; |
| longest = str.substring(start, end + 1); |
| } |
| } |
| return longest; |
| } |
|  |
| // Example |
| String str = "abcabcbb"; |
| System.out.println(longestUniqueSubstring(str)); // Output: abc |
| 96. Calculate the square root of a number: |
| code |
|  |
| double calculateSquareRoot(double num) { |
| return Math.sqrt(num); |
| } |
|  |
| // Example |
| double num = 16; |
| System.out.println(calculateSquareRoot(num)); // Output: 4.0 |
| 97. Find the total number of prime numbers less than a given number: |
| code |
|  |
| int countPrimesLessThan(int num) { |
| int count = 0; |
| for (int i = 2; i < num; i++) { |
| if (isPrime(i)) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| boolean isPrime(int num) { |
| if (num <= 1) return false; |
| for (int i = 2; i <= Math.sqrt(num); i++) { |
| if (num % i == 0) return false; |
| } |
| return true; |
| } |
|  |
| // Example |
| int num = 10; |
| System.out.println(countPrimesLessThan(num)); // Output: 4 |
| 98. Check if a string is a valid URL: |
| code |
|  |
| boolean isValidURL(String str) { |
| try { |
| new code.net.URL(str).toURI(); |
| return true; |
| } catch (code.net.MalformedURLException | code.net.URISyntaxException e) { |
| return false; |
| } |
| } |
|  |
| // Example |
| String url = "https://www.example.com"; |
| System.out.println(isValidURL(url)); // Output: true |
| 99. Count the number of vowels in a string: |
| code |
|  |
| int countVowels(String str) { |
| int count = 0; |
| for (char c : str.toLowerCase().toCharArray()) { |
| if ("aeiou".indexOf(c) != -1) { |
| count++; |
| } |
| } |
| return count; |
| } |
|  |
| // Example |
| String str = "Hello World"; |
| System.out.println(countVowels(str)); // Output: 3 |
| 100. Find the intersection of two sets: |
| code |
|  |
| code.util.Set<Integer> intersection(code.util.Set<Integer> set1, code.util.Set<Integer> set2) { |
| code.util.Set<Integer> result = new code.util.HashSet<>(set1); |
| result.retainAll(set2); |
| return result; |
| } |
|  |
| // Example |
| code.util.Set<Integer> set1 = code.util.Set.of(1, 2, 3, 4); |
| code.util.Set<Integer> set2 = code.util.Set.of(3, 4, 5, 6); |
| System.out.println(intersection(set1, set2)); // Output: [3, 4] |

**Provide the logical problem**: Describe a common real-world scenario.

1. **Show the code solution**: Provide the code that solves the problem.
2. **Explanation**: Briefly explain how the logic works.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No** | | **Logical Problem Description** | | **Code Example** | | | **Explanation** | |
| 1 | | Check if a number is prime | | boolean isPrime = true; for (int i = 2; i <= num / 2; i++) { if (num % i == 0) { isPrime = false; break; } } | | | Loops through possible divisors to check if the number has any divisors other than 1 and itself. | |
| 2 | | Reverse a string | | String rev = new StringBuilder(str).reverse().toString(); | | | Uses StringBuilder to reverse a string. | |
| 3 | | Check if a string is a palindrome | | String rev = new StringBuilder(str).reverse().toString(); if (str.equals(rev)) | | | Compares a string with its reverse to check if it’s a palindrome. | |
| 4 | | Count occurrences of a character in a string | | long count = str.chars().filter(ch -> ch == 'a').count(); | | | Filters and counts the occurrences of a specific character. | |
| 5 | | Find the largest of three numbers | | int max = Math.max(num1, Math.max(num2, num3)); | | | Uses Math.max to find the largest among three numbers. | |
| 6 | | Check if a number is a perfect square | | if (Math.sqrt(num) == (int)Math.sqrt(num)) | | | Checks if the square root of a number is an integer. | |
| 7 | | Find the GCD of two numbers | | while (b != 0) { int t = b; b = a % b; a = t; } | | | Implements the Euclidean algorithm to find the greatest common divisor. | |
| 8 | | Find the LCM of two numbers | | int lcm = (num1 \* num2) / gcd(num1, num2); | | | Uses the relationship between GCD and LCM to find the least common multiple. | |
| 9 | | Generate Fibonacci sequence up to n terms | | int[] fib = new int[n]; fib[0] = 0; fib[1] = 1; for (int i = 2; i < n; i++) { fib[i] = fib[i-1] + fib[i-2]; } | | | Uses an array to generate the Fibonacci sequence. | |
| 10 | | Calculate the sum of digits of a number | | int sum = 0; while (num > 0) { sum += num % 10; num /= 10; } | | | Extracts and sums each digit of a number. | |
| 11 | | Find the second largest element in an array | | Arrays.sort(arr); int secondLargest = arr[arr.length - 2]; | | | Sorts the array and retrieves the second last element. | |
| 12 | | Find the factorial of a number using recursion | | int factorial(int n) { if (n == 0) return 1; return n \* factorial(n - 1); } | | | Recursively calculates the factorial of a number. | |
| 13 | | Check if two strings are anagrams | | char[] a = str1.toCharArray(); char[] b = str2.toCharArray(); Arrays.sort(a); Arrays.sort(b); return Arrays.equals(a, b); | | | Sorts and compares two strings to check if they are anagrams. | |
| 14 | | Find the maximum sum subarray (Kadane's Algorithm) | | int maxSoFar = arr[0], maxEndingHere = arr[0]; for (int i = 1; i < arr.length; i++) { maxEndingHere = Math.max(arr[i], maxEndingHere + arr[i]); maxSoFar = Math.max(maxSoFar, maxEndingHere); } | | | Implements Kadane's Algorithm to find the maximum sum of a contiguous subarray. | |
| 15 | | Convert a binary number to decimal | | int decimal = Integer.parseInt(binaryStr, 2); | | | Converts a binary string to a decimal integer. | |
| 16 | | Convert a decimal number to binary | | String binary = Integer.toBinaryString(decimal); | | | Converts a decimal integer to a binary string. | |
| 17 | | Find all prime numbers up to a given number | | boolean[] prime = new boolean[n + 1]; Arrays.fill(prime, true); for (int p = 2; p \* p <= n; p++) { if (prime[p]) { for (int i = p \* p; i <= n; i += p) prime[i] = false; } } | | | Uses the Sieve of Eratosthenes to find all prime numbers up to a given number. | |
| 18 | | Find the longest common prefix in an array of strings | | String prefix = strs[0]; for (int i = 1; i < strs.length; i++) { while (strs[i].indexOf(prefix) != 0) { prefix = prefix.substring(0, prefix.length() - 1); } } | | | Iteratively reduces the prefix until it matches all strings. | |
| 19 | | Remove duplicates from an array | | int[] unique = Arrays.stream(arr).distinct().toArray(); | | | Uses Streams to remove duplicates from an array. | |
| 20 | | Rotate an array to the right by n positions | | Collections.rotate(Arrays.asList(arr), n); | | | Rotates an array to the right by a given number of positions. | |
| 21 | | Check if a string contains only digits | | boolean onlyDigits = str.chars().allMatch(Character::isDigit); | | | Checks if a string contains only numeric characters. | |
| 22 | | Find the missing number in a series of consecutive numbers | | int sum = n \* (n + 1) / 2; int actualSum = Arrays.stream(arr).sum(); int missingNumber = sum - actualSum; | | | Calculates the difference between the expected sum and actual sum to find the missing number. | |
| 23 | | Check if a number is a palindrome | | int reversed = 0, original = num; while (num != 0) { int digit = num % 10; reversed = reversed \* 10 + digit; num /= 10; } if (original == reversed) | | | Reverses the number and compares it with the original. | |
| 24 | | Find the sum of elements in an array | | int sum = Arrays.stream(arr).sum(); | | | Sums all elements in an array using Streams. | |
| 25 | | Find the common elements between two arrays | | int[] common = Arrays.stream(arr1).filter(x -> Arrays.stream(arr2).anyMatch(y -> y == x)).toArray(); | | | Finds common elements between two arrays using Streams. | |
| 26 | | Find the maximum product of two integers in an array | | Arrays.sort(arr); int maxProduct = arr[arr.length - 1] \* arr[arr.length - 2]; | | | Sorts the array and multiplies the two largest elements. | |
| 27 | | Convert a string to an array of words | | String[] words = str.split("\\s+"); | | | Splits a string into an array of words based on whitespace. | |
| 28 | | Check if an array is sorted | | boolean sorted = IntStream.range(0, arr.length - 1).allMatch(i -> arr[i] <= arr[i + 1]); | | | Checks if each element in an array is less than or equal to the next. | |
| 29 | | Merge two sorted arrays | | int[] merged = IntStream.concat(Arrays.stream(arr1), Arrays.stream(arr2)).sorted().toArray(); | | | Merges and sorts two arrays using Streams. | |
| 30 | | Find the median of an array | | Arrays.sort(arr); double median; int n = arr.length; if (n % 2 == 0) median = (arr[n/2] + arr[n/2 - 1]) / 2.0; else median = arr[n/2]; | | | Sorts the array and finds the middle value or average of the middle values. | |
| 31 | | Find the first non-repeated character in a string | | char firstNonRepeated = str.chars().mapToObj(i -> (char) i).filter(ch -> str.indexOf(ch) == str.lastIndexOf(ch)).findFirst().orElse('\_'); | | | Finds the first character that does not repeat in a string. | |
| 32 | | Convert a string to title case | | String titleCase = Arrays.stream(str.split("\\s+")).map(word -> Character.toUpperCase(word.charAt(0)) + word.substring(1).toLowerCase()).collect(Collectors.joining(" ")); | | | Converts each word to title case (first letter uppercase, rest lowercase). | |
| 33 | | Find all permutations of a string | | permute(str, 0, str.length() - 1); | | | Uses recursion to generate all permutations of a string. | |
| 34 | | Check if a string is a valid email | | boolean isValid = str.matches("^[A-Za-z0-9+\_.-]+@(.+)$"); | | | Uses a regular expression to validate an email format. | |
| 35 | | Calculate the power of a number | | double result = Math.pow(base, exponent); | | | Calculates the result of a number raised to a power. | |
| 36 | | Convert a decimal number to hexadecimal | | String hex = Integer.toHexString(num); | | | Converts a decimal integer to a hexadecimal string. | |
| 37 | | Convert a string to an integer | | int num = Integer.parseInt(str); | | | Parses a string to return an integer value. | |
| 38 | | Convert a string to a double | | double num = Double.parseDouble(str); | | | Parses a string to return a double value. | |
| 39 | | Convert a double to a string | | String str = Double.toString(num); | | | Converts a double value to a string. | |
| 40 | | Count the number of vowels in a string | | long count = str.chars().filter(c -> "AEIOUaeiou".indexOf(c) != -1).count(); | | | Filters and counts the vowels in a string. | |
| 41 | | Remove all whitespace from a string | | String noSpaces = str.replaceAll("\\s", ""); | | | Removes all spaces, tabs, and newlines from a string. | |
| 42 | | Remove all non-alphanumeric characters | | String alphanumeric = str.replaceAll("[^A-Za-z0-9]", ""); | | | Removes all characters that are not letters or digits. | |
| 43 | | Find the longest word in a sentence | | String longest = Arrays.stream(str.split("\\s+")).max(Comparator.comparingInt(String::length)).orElse(""); | | | Finds the longest word in a sentence. | |
| 44 | | Sort an array of strings by length | | Arrays.sort(arr, Comparator.comparingInt(String::length)); | | | Sorts strings in an array by their length. | |
| 45 | | Find the intersection of two lists | | List<Integer> intersection = list1.stream().filter(list2::contains).collect(Collectors.toList()); | | | Finds common elements between two lists. | |
| 46 | | Remove duplicates from a list | | List<Integer> unique = list.stream().distinct().collect(Collectors.toList()); | | | Removes duplicate elements from a list. | |
| 47 | | Calculate the average of an array | | double average = Arrays.stream(arr).average().orElse(0); | | | Calculates the average value of an array of numbers. | |
| 48 | | Reverse an array of integers | | Collections.reverse(Arrays.asList(arr)); | | | Reverses the elements of an array. | |
| 49 | | Count the frequency of each character in a string | | Map<Character, Long> freq = str.chars().mapToObj(c -> (char)c).collect(Collectors.groupingBy(c -> c, Collectors.counting())); | | | Counts how many times each character appears in a string. | |
| 50 | | Merge two lists and remove duplicates | | List<Integer> merged = Stream.concat(list1.stream(), list2.stream()).distinct().collect(Collectors.toList()); | | | Merges two lists and removes any duplicate elements. | |
| 51 | | Find the mode (most frequent element) in an array | | int mode = Arrays.stream(arr).boxed().collect(Collectors.groupingBy(e -> e, Collectors.counting())).entrySet().stream().max(Map.Entry.comparingByValue()).get().getKey(); | | | Finds the most frequent element in an array. | |
| 52 | | Remove the last element from a list | | list.remove(list.size() - 1); | | | Removes the last item from a list. | |
| 53 | | Find the longest substring without repeating characters | | String result = ""; for (int i = 0; i < str.length(); i++) { for (int j = i + 1; j <= str.length(); j++) { String sub = str.substring(i, j); if (sub.chars().distinct().count() == sub.length()) { result = result.length() < sub.length() ? sub : result; } } } | | | Finds the longest substring with all unique characters. | |
| 54 | | Calculate the sum of squares of an array | | int sum = Arrays.stream(arr).map(i -> i \* i).sum(); | | | Calculates the sum of the squares of each element in an array. | |
| 55 | | Convert a number to its ordinal representation | | String ordinal = num + (num % 100 >= 11 && num % 100 <= 13 ? "th" : num % 10 == 1 ? "st" : num % 10 == 2 ? "nd" : num % 10 == 3 ? "rd" : "th"); | | | Converts a number into its ordinal form (1st, 2nd, 3rd, etc.). | |
| 56 | | Check if a number is in a given range | | boolean inRange = num >= lower && num <= upper; | | | Checks if a number falls within a specific range. | |
| 57 | | Find the average word length in a sentence | | double avgLength = Arrays.stream(str.split("\\s+")).mapToInt(String::length).average().orElse(0); | | | Calculates the average length of words in a sentence. | |
| 58 | | Capitalize the first letter of each word | | String capitalized = Arrays.stream(str.split("\\s+")).map(w -> Character.toUpperCase(w.charAt(0)) + w.substring(1)).collect(Collectors.joining(" ")); | | | Capitalizes the first letter of each word in a sentence. | |
| 59 | | Find the sum of all even numbers in an array | | int sumEven = Arrays.stream(arr).filter(i -> i % 2 == 0).sum(); | | | Sums up all the even numbers in an array. | |
| 60 | | Find the sum of all odd numbers in an array | | int sumOdd = Arrays.stream(arr).filter(i -> i % 2 != 0).sum(); | | | Sums up all the odd numbers in an array. | |
| 61 | | Generate the first n prime numbers | | List<Integer> primes = new ArrayList<>(); int num = 2; while (primes.size() < n) { boolean isPrime = true; for (int i = 2; i <= Math.sqrt(num); i++) { if (num % i == 0) { isPrime = false; break; } } if (isPrime) primes.add(num); num++; } | | | Generates the first n prime numbers. | |
| 62 | | Calculate the nth Fibonacci number using recursion | | int fib(int n) { if (n <= 1) return n; return fib(n - 1) + fib(n - 2); } | | | Recursively calculates the nth Fibonacci number. | |
| 63 | | Find the index of the first occurrence of a substring | | int index = str.indexOf(sub); | | | Finds the index of the first occurrence of a substring in a string. | |
| 64 | | Calculate the product of elements in an array | | int product = Arrays.stream(arr).reduce(1, (a, b) -> a \* b); | | | Calculates the product of all elements in an array. | |
| 65 | | Calculate the factorial of a number using iteration | | int factorial = 1; for (int i = 2; i <= num; i++) factorial \*= i; | | | Iteratively calculates the factorial of a number. | |
| 66 | | Check if a number is a power of two | | boolean isPowerOfTwo = (num & (num - 1)) == 0; | | | Uses bitwise operations to check if a number is a power of two. | |
| 67 | | Count the number of set bits in an integer | | int count = Integer.bitCount(num); | | | Counts the number of 1s in the binary representation of an integer. | |
| 68 | | Check if an array is a palindrome | | boolean isPalindrome = IntStream.range(0, arr.length / 2).allMatch(i -> arr[i] == arr[arr.length - i - 1]); | | | Checks if an array reads the same forward and backward. | |
| 69 | | Convert a list of integers to a comma-separated string | | String result = list.stream().map(String::valueOf).collect(Collectors.joining(",")); | | | Converts a list of integers into a string with elements separated by commas. | |
| 70 | | Find the maximum depth of a binary tree | | int maxDepth(TreeNode node) { if (node == null) return 0; return 1 + Math.max(maxDepth(node.left), maxDepth(node.right)); } | | | Recursively calculates the maximum depth of a binary tree. | |
| 71 | | Implement a binary search algorithm | | int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = left + (right - left) / 2; if (arr[mid] == target) return mid; if (arr[mid] < target) left = mid + 1; else right = mid - 1; } return -1; } | | | Implements binary search to find a target value in a sorted array. | |
| 72 | | Implement the merge sort algorithm | | `void mergeSort(int[] arr, int l, int r) { if (l < r) { int m = (l + r) / 2; mergeSort(arr, l, m); mergeSort(arr, m + 1, r); merge(arr, l, m, r); } } void merge(int[] arr, int l, int m, int r) { int n1 | | |  | |
| 73 | Check if two strings are anagrams | | | boolean isAnagram = Arrays.equals(str1.chars().sorted().toArray(), str2.chars().sorted().toArray()); | | Checks if two strings contain the same characters in a different order. | | |
| 74 | Generate all subsets of a set | | | void generateSubsets(int[] arr, int index, List<Integer> subset) { if (index == arr.length) { System.out.println(subset); return; } generateSubsets(arr, index + 1, subset); subset.add(arr[index]); generateSubsets(arr, index + 1, subset); subset.remove(subset.size() - 1); } | | Recursively generates all subsets of a given set. | | |
| 75 | Rotate an array to the right by k steps | | | void rotate(int[] nums, int k) { k %= nums.length; reverse(nums, 0, nums.length - 1); reverse(nums, 0, k - 1); reverse(nums, k, nums.length - 1); } | | Rotates an array to the right by k steps using array reversal. | | |
| 76 | Check if a string contains only digits | | | boolean isNumeric = str.matches("\\d+"); | | Checks if a string contains only numeric digits. | | |
| 77 | Find the greatest common divisor (GCD) of two numbers | | | int gcd(int a, int b) { return b == 0 ? a : gcd(b, a % b); } | | Recursively finds the greatest common divisor (GCD) of two numbers using the Euclidean algorithm. | | |
| 78 | Check if a string is a valid palindrome | | | boolean isPalindrome = str.equals(new StringBuilder(str).reverse().toString()); | | Checks if a string reads the same forward and backward. | | |
| 79 | Convert a binary string to a decimal number | | | int decimal = Integer.parseInt(binaryStr, 2); | | Converts a binary string to its decimal equivalent. | | |
| 80 | Implement the quicksort algorithm | | | void quickSort(int[] arr, int low, int high) { if (low < high) { int pi = partition(arr, low, high); quickSort(arr, low, pi - 1); quickSort(arr, pi + 1, high); } } int partition(int[] arr, int low, int high) { int pivot = arr[high]; int i = (low - 1); for (int j = low; j < high; j++) { if (arr[j] <= pivot) { i++; int temp = arr[i]; arr[i] = arr[j]; arr[j] = temp; } } int temp = arr[i + 1]; arr[i + 1] = arr[high]; arr[high] = temp; return i + 1; } | | Implements the quicksort algorithm to sort an array of integers. | | |
| 81 | Find the sum of all prime numbers up to a given number | | | int sumPrimes = IntStream.rangeClosed(2, n).filter(this::isPrime).sum(); | | Calculates the sum of all prime numbers up to a given number. | | |
| 82 | Check if a number is prime | | | boolean isPrime(int n) { if (n <= 1) return false; for (int i = 2; i <= Math.sqrt(n); i++) if (n % i == 0) return false; return true; } | | Checks if a number is prime by testing divisibility. | | |
| 83 | Convert a decimal number to binary | | | String binary = Integer.toBinaryString(num); | | Converts a decimal integer to its binary string representation. | | |
| 84 | Find the minimum element in a rotated sorted array | | | int findMin(int[] nums) { int left = 0, right = nums.length - 1; while (left < right) { int mid = (left + right) / 2; if (nums[mid] > nums[right]) left = mid + 1; else right = mid; } return nums[left]; } | | Finds the minimum element in a rotated sorted array using binary search. | | |
| 85 | Implement a depth-first search (DFS) on a graph | | | void dfs(Node node) { if (node == null) return; visited.add(node); for (Node neighbor : node.neighbors) if (!visited.contains(neighbor)) dfs(neighbor); } | | Implements depth-first search (DFS) on a graph to explore all nodes and edges. | | |
| 86 | Implement a breadth-first search (BFS) on a graph | | | void bfs(Node start) { Queue<Node> queue = new LinkedList<>(); queue.add(start); visited.add(start); while (!queue.isEmpty()) { Node node = queue.poll(); for (Node neighbor : node.neighbors) if (!visited.contains(neighbor)) { visited.add(neighbor); queue.add(neighbor); } } } | | Implements breadth-first search (BFS) on a graph to explore all nodes level by level. | | |
| 87 | Find the nth root of a number | | | double nthRoot = Math.pow(num, 1.0 / n); | | Calculates the nth root of a number using the power function. | | |
| 88 | Calculate the median of an array | | | double median; Arrays.sort(arr); if (arr.length % 2 == 0) median = ((double)arr[arr.length/2] + (double)arr[arr.length/2 - 1])/2; else median = (double) arr[arr.length/2]; | | Finds the median value of an array of numbers. | | |
| 89 | Convert a list of strings to uppercase | | | List<String> upperCaseList = list.stream().map(String::toUpperCase).collect(Collectors.toList()); | | Converts all strings in a list to uppercase. | | |
| 90 | Find the maximum sum subarray (Kadane's algorithm) | | | int maxSubArray(int[] nums) { int maxSoFar = nums[0], maxEndingHere = nums[0]; for (int i = 1; i < nums.length; i++) { maxEndingHere = Math.max(nums[i], maxEndingHere + nums[i]); maxSoFar = Math.max(maxSoFar, maxEndingHere); } return maxSoFar; } | | Finds the contiguous subarray with the maximum sum in an array using Kadane's algorithm. | | |
| 91 | Reverse words in a sentence | | | String reversed = Arrays.stream(str.split("\\s+")).map(word -> new StringBuilder(word).reverse().toString()).collect(Collectors.joining(" ")); | | Reverses each word in a sentence. | | |
| 92 | Check if a string is a valid IP address | | | `boolean isValidIP = str.matches("\b((25[0-5] | | 2[0-4][0-9] | | |
| 93 | Check if a number is a perfect square | | | boolean isPerfectSquare = Math.sqrt(num) % 1 == 0; | | Checks if a number is a perfect square. | | |
| 94 | Find the least common multiple (LCM) of two numbers | | | int lcm(int a, int b) { return a \* (b / gcd(a, b)); } | | Calculates the least common multiple (LCM) of two numbers using the GCD. | | |
| 95 | Calculate the perimeter of a rectangle | | | int perimeter = 2 \* (length + width); | | Calculates the perimeter of a rectangle given its length and width. | | |
| 96 | Implement a basic calculator | | | `int result = eval(expression); int eval(String expression) { Stack<Integer> stack = new Stack<>(); int num = 0; char sign = '+'; for (int i = 0; i < expression.length(); i++) { char c = expression.charAt(i); if (Character.isDigit(c)) { num = num \* 10 + (c - '0'); } if (!Character.isDigit(c) && !Character.isWhitespace(c) | |  | | |
| |  | | --- | | 97 | | |  | | --- | | Check if two  are rotations of each other | | | | |  | | --- | | boolean areRotations = (str1.length() == str2.length()) && (str1 + str1).contains(str2); | | | |  | | --- | | Checks if one string is a rotation of another by concatenating the first string with itself and checking if the second string is a substring of the result. | | | |
| 98 | Find the intersection of two arrays | | int[] intersection = Arrays.stream(arr1).distinct().filter(x -> Arrays.stream(arr2).anyMatch(y -> y == x)).toArray(); | | Finds the common elements between two arrays. | | |
| 99 | Find the missing number in an array | | int missingNumber = IntStream.rangeClosed(1, n).sum() - Arrays.stream(arr).sum(); | | Finds the missing number in a sequence of 1 to n by subtracting the sum of the array from the sum of the complete sequence. | | |
| 100 | Check if a string is a valid email address | | boolean isValidEmail = email.matches("^[A-Za-z0-9+\_.-]+@[A-Za-z0-9.-]+$"); | | Uses a regular expression to validate if a string is a valid email address. | | |
|  |  | |  | |  | | |