# Закрепление массивов, Java-Basic

1. Return the number of even ints in the given array. Note: the % "mod" operator computes the remainder, e.g. 5 % 2 is 1.

```
countEvens([2, 1, 2, 3, 4]) → 3
countEvens([2, 2, 0]) → 3
countEvens([1, 3, 5]) → 0
```

2. Return the sum of the numbers in the array, returning 0 for an empty array. Except the number 13 is very unlucky, so it does not count and numbers that come immediately after a 13 also do not count.

```
sum13([1, 2, 2, 1]) → 6
sum13([1, 1]) → 2
sum13([1, 2, 2, 1, 13]) → 6
```

3. Given an array of ints, return true if the array contains no 1's and no 3's.

```
lucky13([0, 2, 4]) → true
lucky13([1, 2, 3]) → false
lucky13([1, 2, 4]) → false
```

4. Given a number n, create and return a new int array of length n, containing the numbers 0, 1, 2, ... n-1. The given n may be 0, in which case just return a length 0 array. You do not need a separate if-statement for the length-0 case; the for-loop should naturally execute 0 times in that case, so it just works. The syntax to make a new int array is: new int[desired_length] (See also: FizzBuzz Code)

```
fizzArray(4) → [0, 1, 2, 3]
fizzArray(1) → [0]
fizzArray(10) → [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

5. Given an array of ints, return true if it contains no 1's or it contains no 4's.

```
no14([1, 2, 3]) → true
no14([1, 2, 3, 4]) → false
no14([2, 3, 4]) → true
```

6. Given arrays nums1 and nums2 of the same length, for every element in nums1, consider the corresponding element in nums2 (at the same index). Return the count of the number of times that the two elements differ by 2 or less, but are not equal.

```
matchUp([1, 2, 3], [2, 3, 10]) → 2
matchUp([1, 2, 3], [2, 3, 5]) → 3
matchUp([1, 2, 3], [2, 3, 3]) → 2
```

7. Given an array of ints, return true if the array contains either 3 even or 3 odd values all next to each other.

```
modThree([2, 1, 3, 5]) → true
modThree([2, 1, 2, 5]) → false
modThree([2, 4, 2, 5]) → true
```

8. Return true if the group of N numbers at the start and end of the array are the same. For example, with {5, 6, 45, 99, 13, 5, 6}, the ends are the same for n=0 and n=2, and false for n=1 and n=3. You may assume that n is in the range 0..nums.length inclusive.

```
sameEnds([5, 6, 45, 99, 13, 5, 6], 1) → false
sameEnds([5, 6, 45, 99, 13, 5, 6], 2) → true
sameEnds([5, 6, 45, 99, 13, 5, 6], 3) → false
```

9. Return an array that is "left shifted" by one -- so {6, 2, 5, 3} returns {2, 5, 3, 6}. You may modify and return the given array, or return a new array.

```
shiftLeft([6, 2, 5, 3]) → [2, 5, 3, 6]
shiftLeft([1, 2]) → [2, 1]
shiftLeft([1]) → [1]
```

10. Given a non-empty array of ints, return a new array containing the elements from the original array that come after the last 4 in the original array. The original array will contain at least one 4. Note that it is valid in java to create an array of length 0.

```
post4([2, 4, 1, 2]) → [1, 2]
post4([4, 1, 4, 2]) → [2]
post4([4, 4, 1, 2, 3]) → [1, 2, 3]
```

11. Return a version of the given array where all the 10's have been removed. The remaining elements should shift left towards the start of the array as needed, and the empty spaces a the end of the array should be 0. So {1, 10, 10, 2} yields {1, 2, 0, 0}. You may modify and return the given array or make a new array.

```
withoutTen([1, 10, 10, 2]) → [1, 2, 0, 0]
withoutTen([10, 2, 10]) → [2, 0, 0]
withoutTen([1, 99, 10]) → [1, 99, 0]
```

12. This is slightly more difficult version of the famous FizzBuzz problem which is sometimes given as a first problem for job interviews. (See also: FizzBuzz Code.) Consider the series of numbers beginning at start and running up to but not including end, so for example start=1 and end=5 gives the series 1, 2, 3, 4. Return a new String[] array containing the string form of these numbers, except for multiples of 3, use "Fizz" instead of the number, for multiples of 5 use "Buzz", and for multiples of both 3 and 5 use "FizzBuzz". In Java, String.valueOf(xxx) will make the String form of an int or other type. This version is a little more complicated than the usual version since you have to allocate and index into an array instead of just printing, and we vary the start/end instead of just always doing 1..100. //02-12---Это немного более сложная версия известной проблемы FizzBuzz, // которую иногда называют первой проблемой для собеседований. (См. Также: Код FizzBuzz.) // Рассмотрим последовательность чисел, начинающуюся с начала и продолжающуюся до, // но не включающую конец, поэтому, например, start = 1 и end = 5 дают серии 1, 2, 3, 4. // Возвращаем новую строку [] массив, содержащий строковую форму этих чисел, // за исключением кратных 3, используйте «Fizz» вместо числа, для кратных 5 используйте «Buzz», // а для кратных 3 и 5 используйте «FizzBuzz». В Java String.valueOf (xxx) // сделает строковую форму типа int или другого типа. Эта версия немного сложнее, чем обычная версия, // так как вам нужно выделять и индексировать массив, // а не просто печатать, и мы меняем начало / конец вместо того, чтобы всегда делать 1..100.

```
fizzBuzz(1, 6) → ["1", "2", "Fizz", "4", "Buzz"]
fizzBuzz(1, 8) → ["1", "2", "Fizz", "4", "Buzz", "Fizz", "7"]
fizzBuzz(1, 11) → ["1", "2", "Fizz", "4", "Buzz", "Fizz", "7", "8", "Fizz", "Buzz"]
```

13. Given an array length 1 or more of ints, return the difference between the largest and smallest values in the array. Note: the built-in Math.min(v1, v2) and Math.max(v1, v2) methods return the smaller or larger of two values.

```
bigDiff([10, 3, 5, 6]) → 7
bigDiff([7, 2, 10, 9]) → 8
bigDiff([2, 10, 7, 2]) → 8
```

14. Return the sum of the numbers in the array, except ignore sections of numbers starting with a 6 and extending to the next 7 (every 6 will be followed by at least one 7). Return 0 for no numbers.

```
sum67([1, 2, 2]) → 5
sum67([1, 2, 2, 6, 99, 99, 7]) → 5
sum67([1, 1, 6, 7, 2]) → 4
```

15. Given an array of ints, return true if the sum of all the 2's in the array is exactly 8.

```
sum28([2, 3, 2, 2, 4, 2]) → true
sum28([2, 3, 2, 2, 4, 2, 2]) → false
sum28([1, 2, 3, 4]) → false
```

16. Given an array of ints, return true if every element is a 1 or a 4.

```
only14([1, 4, 1, 4]) → true
only14([1, 4, 2, 4]) → false
only14([1, 1]) → true
```

17. We'll say that a value is "everywhere" in an array if for every pair of adjacent elements in the array, at least one of the pair is that value. Return true if the given value is everywhere in the array.

```
isEverywhere([1, 2, 1, 3], 1) → true
isEverywhere([1, 2, 1, 3], 2) → false
isEverywhere([1, 2, 1, 3, 4], 1) → false
```

18. Given an array of ints, return true if the array contains two 7's next to each other, or there are two 7's separated by one element, such as with {7, 1, 7}.

```
has77([1, 7, 7]) → true
has77([1, 7, 1, 7]) → true
has77([1, 7, 1, 1, 7]) → false
```

19. Given an array of ints, return true if the value 3 appears in the array exactly 3 times, and no 3's are next to each other.

```
```

```
haveThree([3, 1, 3, 1, 3]) → true
haveThree([3, 1, 3, 3]) → false
haveThree([3, 4, 3, 3, 4]) → false
```

20. Return true if the array contains, somewhere, three increasing adjacent numbers like .... 4, 5, 6, ... or 23, 24, 25.

```
tripleUp([1, 4, 5, 6, 2]) → true
tripleUp([1, 2, 3]) → true
tripleUp([1, 2, 4]) → false
```

21. For each multiple of 10 in the given array, change all the values following it to be that multiple of 10, until encountering another multiple of 10. So {2, 10, 3, 4, 20, 5} yields {2, 10, 10, 10, 20, 20}.

```
tenRun([2, 10, 3, 4, 20, 5]) → [2, 10, 10, 10, 20, 20]
tenRun([10, 1, 20, 2]) → [10, 10, 20, 20]
tenRun([10, 1, 9, 20]) → [10, 10, 10, 20]
```

22. We'll say that an element in an array is "alone" if there are values before and after it, and those values are different from it. Return a version of the given array where every instance of the given value which is alone is replaced by whichever value to its left or right is larger.

```
notAlone([1, 2, 3], 2) → [1, 3, 3]
notAlone([1, 2, 3, 2, 5, 2], 2) → [1, 3, 3, 5, 5, 2]
notAlone([3, 4], 3) → [3, 4]
```

23. Return a version of the given array where each zero value in the array is replaced by the largest odd value to the right of the zero in the array. If there is no odd value to the right of the zero, leave the zero as a zero.

```
zeroMax([0, 5, 0, 3]) → [5, 5, 3, 3]
zeroMax([0, 4, 0, 3]) → [3, 4, 3, 3]
zeroMax([0, 1, 0]) → [1, 1, 0]
```

24. Return the "centered" average of an array of ints, which we'll say is the mean average of the values, except ignoring the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. Use int division to produce the final average. You may assume that the array is length 3 or more.

```
centeredAverage([1, 2, 3, 4, 100]) → 3
centeredAverage([1, 1, 5, 5, 10, 8, 7]) → 5
centeredAverage([-10, -4, -2, -4, -2, 0]) → -3
```

25. Given an array of ints, return true if the array contains a 2 next to a 2 somewhere.

```
has22([1, 2, 2]) → true
has22([1, 2, 1, 2]) → false
has22([2, 1, 2]) → false
```

26. Given an array of ints, return true if the number of 1's is greater than the number of 4's

```
more14([1, 4, 1]) → true
more14([1, 4, 1, 4]) → false
more14([1, 1]) → true
```

27. Given a number n, create and return a new string array of length n, containing the strings "0", "1" "2" .. through n-1. N may be 0, in which case just return a length 0 array. Note: String.valueOf(xxx) will make the String form of most types. The syntax to make a new string array is: new String[desired_length] (See also: FizzBuzz Code)

```
fizzArray2(4) → ["0", "1", "2", "3"]
fizzArray2(10) → ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]
fizzArray2(2) → ["0", "1"]
```

28. Given an array of ints, return true if the array contains a 2 next to a 2 or a 4 next to a 4, but not both.

```
either24([1, 2, 2]) → true
either24([4, 4, 1]) → true
either24([4, 4, 1, 2, 2]) → false
```

29. Given an array of ints, return true if there is a 1 in the array with a 2 somewhere later in the array.

```
has12([1, 3, 2]) → true
has12([3, 1, 2]) → true
has12([3, 1, 4, 5, 2]) → true
```

30. Given an array of ints, return true if every 2 that appears in the array is next to another 2.

```
twoTwo([4, 2, 2, 3]) → true
twoTwo([2, 2, 4]) → true
twoTwo([2, 2, 4, 2]) → false
```

31. Given start and end numbers, return a new array containing the sequence of integers from start up to but not including end, so start=5 and end=10 yields {5, 6, 7, 8, 9}. The end number will be greater or equal to the start number. Note that a length-0 array is valid. (See also: FizzBuzz Code)

```
fizzArray3(5, 10) → [5, 6, 7, 8, 9]
fizzArray3(11, 18) → [11, 12, 13, 14, 15, 16, 17]
fizzArray3(1, 3) → [1, 2]
```

32. Given a non-empty array of ints, return a new array containing the elements from the original array that come before the first 4 in the original array. The original array will contain at least one 4. Note that it is valid in java to create an array of length 0.

```
pre4([1, 2, 4, 1]) → [1, 2]
pre4([3, 1, 4]) → [3, 1]
pre4([1, 4, 4]) → [1]
```

33. Return an array that contains the exact same numbers as the given array, but rearranged so that all the zeros are grouped at the start of the array. The order of the non-zero numbers does not matter. So {1, 0, 0, 1} becomes {0 ,0, 1, 1}. You may modify and return the given array or make a new array.

```
zeroFront([1, 0, 0, 1]) → [0, 0, 1, 1]
zeroFront([0, 1, 1, 0, 1]) → [0, 0, 1, 1, 1]
zeroFront([1, 0]) → [0, 1]
```

34. Return an array that contains the exact same numbers as the given array, but rearranged so that all the even numbers come before all the odd numbers. Other than that, the numbers can be in any order. You may modify and return the given array, or make a new array.

```
evenOdd([1, 0, 1, 0, 0, 1, 1]) → [0, 0, 0, 1, 1, 1, 1]
evenOdd([3, 3, 2]) → [2, 3, 3]
evenOdd([2, 2, 2]) → [2, 2, 2]
```