Закрепление массивов, Java-Basic

1. Given an array of ints, return true if 6 appears as either the first or last element in the array. The array will be length 1 or more.

   Дан массив int. Верните true, если цифра 6 является значением в первом или последнем элементе массива. Массив будет длиной больше, либо равной единице.

   ```
   firstLast6([1, 2, 6]) → true
   firstLast6([6, 1, 2, 3]) → true
   firstLast6([13, 6, 1, 2, 3]) → false
   ```

2. Given 2 arrays of ints, a and b, return true if they have the same first element or they have the same last element. Both arrays will be length 1 or more.

   Получив 2 массива int[], a и b, возвращаем **true**, если они имеют один и тот же первый или последний элемент. Оба массива будут длиной больше, либо равной единице.

   ```
   commonEnd([1, 2, 3], [7, 3]) → true
   commonEnd([1, 2, 3], [7, 3, 2]) → false
   commonEnd([1, 2, 3], [1, 3]) → true
   ```

3. Given an array of ints length 3, return a new array with the elements in reverse order, so {1, 2, 3} becomes {3, 2, 1}.

   Дан массив длиной 3, верните новый массив с элементами в обратном порядке, например {1, 2, 3} становится {3, 2, 1}.

   ```
   reverse3([1, 2, 3]) → [3, 2, 1]
   reverse3([5, 11, 9]) → [9, 11, 5]
   reverse3([7, 0, 0]) → [0, 0, 7]
   ```

4. Given 2 int arrays, a and b, each length 3, return a new array length 2 containing their middle elements.

   Даны 2 массива int, a и b, каждый длиной 3, верните новый массив с длиной 2, содержащего среднее арифметическое элементов первоначальных массивов.

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

5. Given an int array length 2, return true if it does not contain a 2 or 3.

   ```
   no23([4, 5]) → true
   no23([4, 2]) → false
   no23([3, 5]) → false
   ```

6. Given an int array length 3, if there is a 2 in the array immediately followed by a 3, set the 3 element to 0. Return the changed array.

```
fix23([1, 2, 3]) → [1, 2, 0]
fix23([2, 3, 5]) → [2, 0, 5]
fix23([1, 2, 1]) → [1, 2, 1]
```

7. Given an array of ints of even length, return a new array length 2 containing the middle two elements from the original array. The original array will be length 2 or more.

```
makeMiddle([1, 2, 3, 4]) → [2, 3]
makeMiddle([7, 1, 2, 3, 4, 9]) → [2, 3]
makeMiddle([1, 2]) → [1, 2]
```

8. Given an array of ints of odd length, return a new array length 3 containing the elements from the middle of the array. The array length will be at least 3.

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

9. We'll say that a 1 immediately followed by a 3 in an array is an "unlucky" 1. Return true if the given array contains an unlucky 1 in the first 2 or last 2 positions in the array.

```
unlucky1([1, 3, 4, 5]) → true
unlucky1([2, 1, 3, 4, 5]) → true
unlucky1([1, 1, 1]) → false
```

10. Given an array of ints, return true if the array is length 1 or more, and the first element and the last element are equal.

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

11. Given an array of ints length 3, return the sum of all the elements.

```
sum3([1, 2, 3]) → 6
sum3([5, 11, 2]) → 18
sum3([7, 0, 0]) → 7
```

12. Given an array of ints length 3, figure out which is larger, the first or last element in the array, and set all the other elements to be that value. Return the changed array.

```
maxEnd3([1, 2, 3]) → [3, 3, 3]
maxEnd3([11, 5, 9]) → [11, 11, 11]
maxEnd3([2, 11, 3]) → [3, 3, 3]
```

13. Given an array of ints, return a new array length 2 containing the first and last elements from the original array. The original array will be length 1 or more.

```
makeEnds([1, 2, 3]) → [1, 3]
makeEnds([1, 2, 3, 4]) → [1, 4]
makeEnds([7, 4, 6, 2]) → [7, 2]
```

14. Given an int array, return a new array with double the length where its last element is the same as the original array, and all the other elements are 0. The original array will be length 1 or more. Note: by default, a new int array contains all 0's.

```
makeLast([4, 5, 6]) → [0, 0, 0, 0, 0, 6]
makeLast([1, 2]) → [0, 0, 0, 2]
makeLast([3]) → [0, 3]
```

15. Start with 2 int arrays, a and b, of any length. Return how many of the arrays have 1 as their first element.

```
start1([1, 2, 3], [1, 3]) → 2
start1([7, 2, 3], [1]) → 1
start1([1, 2], []) → 1
```

16. Given 2 int arrays, each length 2, return a new array length 4 containing all their elements.

```
plusTwo([1, 2], [3, 4]) → [1, 2, 3, 4]
plusTwo([4, 4], [2, 2]) → [4, 4, 2, 2]
plusTwo([9, 2], [3, 4]) → [9, 2, 3, 4]
```

17. Given an array of ints of odd length, look at the first, last, and middle values in the array and return the largest. The array length will be a least 1.

```
maxTriple([1, 2, 3]) → 3
maxTriple([1, 5, 3]) → 5
maxTriple([5, 2, 3]) → 5
```

18. Given 2 int arrays, a and b, return a new array length 2 containing, as much as will fit, the elements from a followed by the elements from b. The arrays may be any length, including 0, but there will be 2 or more elements available between the 2 arrays.

```
make2([4, 5], [1, 2, 3]) → [4, 5]
make2([4], [1, 2, 3]) → [4, 1]
make2([], [1, 2]) → [1, 2]
```

19. Return an int array length 3 containing the first 3 digits of pi, {3, 1, 4}.

```
makePi() → [3, 1, 4]
```

20. Given an array of ints length 3, return an array with the elements "rotated left" so {1, 2, 3} yields {2, 3, 1}.

```
rotateLeft3([1, 2, 3]) → [2, 3, 1]
rotateLeft3([5, 11, 9]) → [11, 9, 5]
rotateLeft3([7, 0, 0]) → [0, 0, 7]
```

21. Given an array of ints, return the sum of the first 2 elements in the array. If the array length is less than 2, just sum up the elements that exist, returning 0 if the array is length 0.

```
sum2([1, 2, 3]) → 3
sum2([1, 1]) → 2
sum2([1, 1, 1, 1]) → 2
```

22. Given an int array length 2, return true if it contains a 2 or a 3.

```
has23([2, 5]) → true
has23([4, 3]) → true
has23([4, 5]) → false
```

23. Given an int array, return true if the array contains 2 twice, or 3 twice. The array will be length 0, 1, or 2.

```
double23([2, 2]) → true
double23([3, 3]) → true
double23([2, 3]) → false
```

24. Start with 2 int arrays, a and b, each length 2. Consider the sum of the values in each array. Return the array which has the largest sum. In event of a tie, return a.

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

25. Given an array of ints, swap the first and last elements in the array. Return the modified array. The array length will be at least 1.

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

26. Given an int array of any length, return a new array of its first 2 elements. If the array is smaller than length 2, use whatever elements are present.

```
frontPiece([1, 2, 3]) → [1, 2]
frontPiece([1, 2]) → [1, 2]
frontPiece([1]) → [1]
```

27. Given 2 int arrays, a and b, of any length, return a new array with the first element of each array. If either array is length 0, ignore that array.

```
front11([1, 2, 3], [7, 9, 8]) → [1, 7]
front11([1], [2]) → [1, 2]
front11([1, 7], []) → [1]
```