|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array-1 | | | | |
| firstLast6 | commonEnd | makeEnds | makeLast | makeMiddle |
| Array-2 | | | | |
| countEvens | bigDiff | centeredAverage | has22 | lucky13 |
| sum28 | more14 | isEverywhere | shiftLeft | post4 |

// 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.  
 public static boolean firstLast6(int[] nums)   
 {

}  
   
 // 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.  
 public static boolean commonEnd(int[] a, int[] b)   
 {

}  
  
 // 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.  
 public int[] makeEnds(int[] nums)   
 {

}  
  
 // 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.   
 // Ex) makeLast([1, 2]) -> [0, 0, 0, 2]  
 public static int[] makeLast(int[] nums)   
 {

}  
 // 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.  
 public static int[] makeMiddle(int[] nums)   
 {

}  
  
 // Return the number of even ints in the given array.   
 public static int countEvens(int[] nums)   
 {

}  
   
 // Given an array length 1 or more of ints,   
 // return the difference between the largest and smallest values in the array.   
 public static int bigDiff(int[] nums)   
 {

}  
  
 // 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.  
 public static int centeredAverage(int[] nums)   
 {

}  
   
 // Given an array of ints, return true if the array contains a 2 next to a 2 somewhere.  
 // Ex) has22([1, 2, 1, 2]) -> false  
 public static boolean has22(int[] nums)   
 {

}  
  
 // Given an array of ints, return true if the array contains no 1's and no 3's.  
 public static boolean lucky13(int[] nums)   
 {

}

// Given an array of ints, return true if the sum of all the 2's in the array is exactly 8.  
 public static boolean sum28(int[] nums)   
 {

}  
  
 // Given an array of ints, return true if the number of 1's is greater than the number of 4's  
 public static boolean more14(int[] nums)   
 {

}  
  
 // 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.  
 // Ex) isEverywhere([1, 2, 1, 2, 3, 2, 5], 2) -> true  
 public static boolean isEverywhere(int[] nums, int val)   
 {

}

// 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.  
 public static int[] shiftLeft(int[] nums) {

}  
  
 // 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.   
 // Ex) post4([4, 1, 4, 2]) -> [2]  
 public static int[] post4(int[] nums)   
 {

}