# Two monkeys

We have two monkeys, a and b, and the parameters aSmile and bSmile indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return true if we are in trouble.   
  
monkeyTrouble(true, true) → true  
monkeyTrouble(false, false) → true  
monkeyTrouble(true, false) → false

public boolean monkeyTrouble(boolean aSmile, boolean bSmile) {

}

# Oz

Given a string, return a string made of the first 2 chars (if present), however include first char only if it is 'o' and include the second only if it is 'z', so "ozymandias" yields "oz".   
  
startOz("ozymandias") → "oz"  
startOz("bzoo") → "z"  
startOz("oxx") → "o"

public String startOz(String str) {

}

# Last 2

Given a string, return the count of the number of times that a substring length 2 appears in the string and also as the last 2 chars of the string, so "hixxxhi" yields 1 (we won't count the end substring).   
  
last2("hixxhi") → 1  
last2("xaxxaxaxx") → 1  
last2("axxxaaxx") → 2

public int last2(String str) {

}

# Array front 9

Given an array of ints, return true if one of the first 4 elements in the array is a 9. The array length may be less than 4.   
  
arrayFront9({1, 2, 9, 3, 4}) → true  
arrayFront9({1, 2, 3, 4, 9}) → false  
arrayFront9({1, 2, 3, 4, 5}) → false

public boolean arrayFront9(int[] nums) {

}

# Color starter

Given a string, if the string begins with "red" or "blue" return that color string, otherwise return the empty string.   
  
seeColor("redxx") → "red"  
seeColor("xxred") → ""  
seeColor("blueTimes") → "blue"

public String seeColor(String str) {

}

# Without start

Given 2 strings, return their concatenation, except omit the first char of each. The strings will be at least length 1.   
  
nonStart("Hello", "There") → "ellohere"  
nonStart("java", "code") → "avaode"  
nonStart("shotl", "java") → "hotlava"

public String nonStart(String a, String b) {

}

# Reverse

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}.   
  
reverse3({1, 2, 3}) → {3, 2, 1}  
reverse3({5, 11, 9}) → {9, 11, 5}  
reverse3({7, 0, 0}) → {0, 0, 7}

public int[] reverse3(int[] nums) {

}

# Unlucky One

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

public boolean unlucky1(int[] nums) {

}

# Great 6

The number 6 is a truly great number. Given two int values, a and b, return true if either one is 6. Or if their sum or difference is 6. Note: the function Math.abs(num) computes the absolute value of a number.   
  
love6(6, 4) → true  
love6(4, 5) → false  
love6(1, 5) → true

public boolean love6(int a, int b) {

}

# Sum limit

Given 2 non-negative ints, a and b, return their sum, so long as the sum has the same number of digits as a. If the sum has more digits than a, just return a without b. (Note: one way to compute the number of digits of a non-negative int n is to convert it to a string with String.valueOf(n) and then check the length of the string.)   
  
sumLimit(2, 3) → 5  
sumLimit(8, 3) → 8  
sumLimit(8, 1) → 9

public int sumLimit(int a, int b) {

}

# Lucky sum

Given 3 int values, a b c, return their sum. However, if one of the values is 13 then it does not count towards the sum and values to its right do not count. So for example, if b is 13, then both b and c do not count.   
  
luckySum(1, 2, 3) → 6  
luckySum(1, 2, 13) → 3  
luckySum(1, 13, 3) → 1

public int luckySum(int a, int b, int c) {

}

# Evenly spaced

Given three ints, a b c, one of them is small, one is medium and one is large. Return true if the three values are evenly spaced, so the difference between small and medium is the same as the difference between medium and large.   
  
evenlySpaced(2, 4, 6) → true  
evenlySpaced(4, 6, 2) → true  
evenlySpaced(4, 6, 3) → false

public boolean evenlySpaced(int a, int b, int c) {

}