1)We want make a package of **goal** kilos of chocolate. We have small bars (1 kilo each) and big bars (5 kilos each). Return the number of small bars to use, assuming we always use big bars before small bars. Return -1 if it can't be done.

makeChocolate(4, 1, 9) → 4

makeChocolate(4, 1, 10) → -1

makeChocolate(4, 1, 7) → 2

public int makeChocolate(int small, int big, int goal) {

}

2) Given a string, return the sum of the digits 0-9 that appear in the string, ignoring all other characters. Return 0 if there are no digits in the string. (Note: Character.isDigit(char) tests if a char is one of the chars '0', '1', .. '9'. Integer.parseInt(string) converts a string to an int.)

sumDigits("aa1bc2d3") → 6

sumDigits("aa11b33") → 8

sumDigits("Chocolate") → 0

public int sumDigits(String str) {

}

3) Given a non-empty array, return true if there is a place to split the array so that the sum of the numbers on one side is equal to the sum of the numbers on the other side.

canBalance([1, 1, 1, 2, 1]) → true

canBalance([2, 1, 1, 2, 1]) → false

canBalance([10, 10]) → true

4) Given a string, compute recursively a new string where all the 'x' chars have been removed.

noX("xaxb") → "ab"

noX("abc") → "abc"

noX("xx") → ""

public String noX(String str) {

}

5) Given two strings, **base** and **remove**, return a version of the base string where all instances of the remove string have been removed (not case sensitive). You may assume that the remove string is length 1 or more. Remove only non-overlapping instances, so with "xxx" removing "xx" leaves "x".

withoutString("Hello there", "llo") → "He there"

withoutString("Hello there", "e") → "Hllo thr"

withoutString("Hello there", "x") → "Hello there"

public String withoutString(String base, String remove) {

}

6) Given n>=0, create an array length n\*n with the following pattern, shown here for n=3 : {0, 0, 1,    0, 2, 1,    3, 2, 1} (spaces added to show the 3 groups).

squareUp(3) → [0, 0, 1, 0, 2, 1, 3, 2, 1]

squareUp(2) → [0, 1, 2, 1]

squareUp(4) → [0, 0, 0, 1, 0, 0, 2, 1, 0, 3, 2, 1, 4, 3, 2, 1]

public int[] squareUp(int n) {

}