Welcome to our first Review Session!

Below is a set of 6 problems. Complete as many as you can. Each problem yields a certain number of points. You get the points for a given problem when you pass all test cases for this problem.

maxTriple [15 pts]

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

squirrelPlay [15 pts]

The squirrels in Palo Alto spend most of the day playing. In particular, they play if the temperature is between 60 and 90 (inclusive). Unless it is summer, then the upper limit is 100 instead of 90. Given an `int temperature` and a `boolean isSummer`, return true if the squirrels play and false otherwise.

squirrelPlay(70, false) → true
squirrelPlay(95, false) → false
squirrelPlay(95, true) → true

maxMod5 [15 pts]

Given two `int` values, return whichever value is larger. However if the two values have the same remainder when divided by 5, then the return the smaller value. However, in all cases, if the two values are the same, return 0. Note: the % "mod" operator computes the remainder, e.g. 7 % 5 is 2.

maxMod5(2, 3) → 3
maxMod5(6, 2) → 6
maxMod5(3, 2) → 3

specialEleven [15 pts]

We'll say a number is special if it is a multiple of 11 or if it is one more than a multiple of 11. Return true if the given non-negative number is special. Use the % "mod" operator.

specialEleven(22) → true
specialEleven(23) → true
specialEleven(24) → false

prefixAgain [12 pts]

Given a `string`, consider the prefix string made of the first N chars of the string. Does that prefix string appear somewhere else in the string? Assume that the string is not empty and that N is in the range 1..str.length().

prefixAgain("abXYabc", 1) → true
prefixAgain("abXYabc", 2) → true
prefixAgain("abXYabc", 3) → false

oneTwo [12 pts]

Given a `string`, compute a new string by moving the first `char` to come after the next two chars, so "abc" yields "bca". Repeat this process for each subsequent group of 3 chars, so "abcdef" yields "bcaefd". Ignore any group of fewer than 3 chars at the end.

oneTwo("abc") → "bca"
oneTwo("tca") → "cat"
oneTwo("tcagdo") → "catdog"

withoutString [8 pts]

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"

sumDigits [8 pts]

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