Recursion

Question - 01

Given a non-negative int n, return the sum of its digits recursively (no loops). Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).

Example:
Input: sumDigits(126)
Output: 9

Input: sumDigits(49)
Output: 13

Input: sumDigits(12)
Output: 3

Question - 02

We have bunnies standing in a line, numbered 1, 2, ... The odd bunnies (1, 3, ..) have the normal 2 ears. The even bunnies (2, 4, ..) we'll say have 3 ears, because they each have a raised foot. Recursively return the number of "ears" in the bunny line 1, 2, ... n (without loops or multiplication).

Example:
Input: bunnyEars2(0)
Output: 0

Input: bunnyEars2(1)
Output: 2

Input: bunnyEars2(2)
Output: 5

Question - 03

Given a non-negative int n, return the count of the occurrences of 7 as a digit, so for example 717 yields 2. (no loops). Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).

Example:
Input: count7(717)
Output: 2

Input: count7(7)
Output: 1

Input: count7(123)
Output: 0

Question - 04

Given a string, compute recursively (no loops) the number of lowercase 'x' chars in the string.

Example:
Input: countX("xxhixx")
Output: 4

Input: countX("xhixhix")
Output: 3

Input: countX("hi")
Output: 0

Question - 05

Given a string, compute recursively (no loops) a new string where all appearances of "pi" have been replaced by "3.14".

Example:
Input: changePi("xpix")
Output: "x3.14x"

Input: changePi("pipi")
Output: "3.143.14"

Input: changePi("pip")
Output: "3.14p"

Question - 06

Given an array of ints, compute recursively the number of times that the value 11 appears in the array. We'll use the convention of considering only the part of the array that begins at the given index. In this way, a recursive call can pass index+1 to move down the array. The initial call will pass in index as 0.

Example:
Input: array11([1, 2, 11], 0)
Output: 1

Input: array11([11, 11], 0)
Output: 2

Input: array11([1, 2, 3, 4], 0)
Output: 0

Question - 07

Given a string, compute recursively a new string where identical chars that are adjacent in the original string are separated from each other by a "*".

Example:
Input: pairStar("hello")
Output: "hel*lo"

Input: pairStar("xxyy")
Output: "x*xy*y"

Input: pairStar("aaaa")
Output: "a*a*a*a"

Question - 08

Count recursively the total number of "abc" and "aba" substrings that appear in the given string.

Example:
Input: countAbc("abc")
Output: 1

Input: countAbc("abcxxabc")
Output: 2

Input: countAbc("abaxxaba")
Output: 2

Question - 09

Given a string, compute recursively the number of times lowercase "hi" appears in the string, however do not count "hi" that have an 'x' immedately before them.

Example:
Input: countHi2("ahixhi")
Output: 1

Input: countHi2("ahibhi")
Output: 2

Input: countHi2("xhixhi")
Output: 0

Question - 10

Given a string and a non-empty substring sub, compute recursively the number of times that sub appears in the string, without the sub strings overlapping.

Example:
Input: strCount("catcowcat", "cat")
Output: 2

Input: strCount("catcowcat", "cow")
Output: 1

Input: strCount("catcowcat", "dog")
Output: 0

Question - 11

We have a number of bunnies and each bunny has two big floppy ears. We want to compute the total number of ears across all the bunnies recursively (without loops or multiplication).

Example:
Input: bunnyEars(0)
Output: 0

Input: bunnyEars(1)
Output: 2

Input: bunnyEars(2)
Output: 4

Question - 12

We have triangle made of blocks. The topmost row has 1 block, the next row down has 2 blocks, the next row has 3 blocks, and so on. Compute recursively (no loops or multiplication) the total number of blocks in such a triangle with the given number of rows.

Example:
Input: triangle(0)
Output: 0

Input: triangle(1)
Output: 1

Input: triangle(2)
Output: 3

Question - 13

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

Example:
Input: noX("xaxb")
Output: "ab"

Input: noX("abc")
Output: "abc"

Input: noX("xx")
Output: ""

Question - 14

Given an array of ints, compute recursively if the array contains somewhere a value followed in the array by that value times 10. We'll use the convention of considering only the part of the array that begins at the given index. In this way, a recursive call can pass index+1 to move down the array. The initial call will pass in index as 0.

Example:
Input: array220([1, 2, 20], 0)
Output: True

Input: array220([3, 30], 0)
Output: True

Input: array220([3], 0)
Output: False

Question - 15

Given a string, compute recursively a new string where all the lowercase 'x' chars have been moved to the end of the string.

Example:
Input: endX("xxre")
Output: "rexx"

Input: endX("xxhixx")
Output: "hixxxx"

Input: endX("xhixhix")
Output: "hihixxx"

Question - 16

Given a string, compute recursively (no loops) the number of "11" substrings in the string. The "11" substrings should not overlap.

Example:
Input: count11("11abc11")
Output: 2

Input: count11("abc11x11x11")
Output: 3

Input: count11("111")
Output: 1

Question - 17

Given a string that contains a single pair of parenthesis, compute recursively a new string made of only of the parenthesis and their contents, so "xyz(abc)123" yields "(abc)".

Example:
Input: parenBit("xyz(abc)123")
Output: "(abc)"

Input: parenBit("x(hello)")
Output: "(hello)"

Input: parenBit("(xy)1")
Output: "(xy)"

Question - 18

Given a string and a non-empty substring sub, compute recursively if at least n copies of sub appear in the string somewhere, possibly with overlapping. N will be non-negative.

Example:
Input: strCopies("catcowcat", "cat", 2)
Output: True

Input: strCopies("catcowcat", "cow", 2)
Output: False

Input: strCopies("catcowcat", "cow", 1)
Output: True

Question - 19

Given a string, compute recursively (no loops) a new string where all the lowercase 'x' chars have been changed to 'y' chars.

Example:
Input: changeXY("codex")
Output: "codey"

Input: changeXY("xxhixx")
Output: "yyhiyy"

Input: changeXY("xhixhix")
Output: "yhiyhiy"

Question - 20

Given an array of ints, compute recursively if the array contains a 6. We'll use the convention of considering only the part of the array that begins at the given index. In this way, a recursive call can pass index+1 to move down the array. The initial call will pass in index as 0.

Example:
Input: array6([1, 6, 4], 0)
Output: True

Input: array6([1, 4], 0)
Output: False

Input: array6([6], 0)
Output: True

Question - 21

Given a string, compute recursively a new string where all the adjacent chars are now separated by a "*".

Example:
Input: allStar("hello")
Output: "h*e*l*l*o"

Input: allStar("abc")
Output: "a*b*c"

Input: allStar("ab")
Output: "a*b"

Question - 22

We'll say that a "pair" in a string is two instances of a char separated by a char. So "AxA" the A's make a pair. Pair's can overlap, so "AxAxA" contains 3 pairs -- 2 for A and 1 for x. Recursively compute the number of pairs in the given string.

Example:
Input: countPairs("axa")
Output: 1

Input: countPairs("axax")
Output: 2

Input: countPairs("axbx")
Output: 1

Question - 23

Given a string, return recursively a "cleaned" string where adjacent chars that are the same have been reduced to a single char. So "yyzzza" yields "yza".

Example:
Input: stringClean("yyzzza")
Output: "yza"

Input: stringClean("abbbcdd")
Output: "abcd"

Input: stringClean("Hello")
Output: "Helo"

Question - 24

Given a string, return true if it is a nesting of zero or more pairs of parenthesis, like "(())" or "((()))". Suggestion: check the first and last chars, and then recur on what's inside them.

Example:
Input: nestParen("(())")
Output: True

Input: nestParen("((()))")
Output: True

Input: nestParen("(((x))")
Output: False

Question - 25

Given a string and a non-empty substring sub, compute recursively the largest substring which starts and ends with sub and return its length.

Example:
Input: strDist("catcowcat", "cat")
Output: 9

Input: strDist("catcowcat", "cow")
Output: 3

Input: strDist("cccatcowcatxx", "cat")
Output: 9

Question - 26

Given an array of ints, is it possible to choose a group of some of the ints, such that the group sums to the given target? This is a classic backtracking recursion problem. Once you understand the recursive backtracking strategy in this problem, you can use the same pattern for many problems to search a space of choices. Rather than looking at the whole array, our convention is to consider the part of the array starting at index start and continuing to the end of the array. The caller can specify the whole array simply by passing start as 0. No loops are needed -- the recursive calls progress down the array.

Example:
Input: groupSum(0, [2, 4, 8], 10)
Output: True

Input: groupSum(0, [2, 4, 8], 14)
Output: True

Input: groupSum(0, [2, 4, 8], 9)
Output: False

Question - 27

Given an array of ints, is it possible to divide the ints into two groups, so that the sums of the two groups are the same. Every int must be in one group or the other. Write a recursive helper method that takes whatever arguments you like, and make the initial call to your recursive helper from splitArray(). (No loops needed.)

Example:
Input: splitArray([2, 2])
Output: True

Input: splitArray([2, 3])
Output: False

Input: splitArray([5, 2, 3])
Output: True

Question - 28

Given an array of ints, is it possible to divide the ints into two groups, so that the sum of one group is a multiple of 10, and the sum of the other group is odd. Every int must be in one group or the other. Write a recursive helper method that takes whatever arguments you like, and make the initial call to your recursive helper from splitOdd10(). (No loops needed.)

Example:
Input: splitOdd10([5, 5, 5])
Output: True

Input: splitOdd10([5, 5, 6])
Output: False

Input: splitOdd10([5, 5, 6, 1])
Output: True

Question - 29

Given an array of ints, is it possible to divide the ints into two groups, so that the sum of the two groups is the same, with these constraints: all the values that are multiple of 5 must be in one group, and all the values that are a multiple of 3 (and not a multiple of 5) must be in the other. (No loops needed.)

Example:
Input: split53([1, 1])
Output: True

Input: split53([1, 1, 1])
Output: False

Input: split53([2, 4, 2])
Output: True

Question - 30

Given an array of ints, is it possible to choose a group of some of the ints, such that the group sums to the given target with these additional constraints: all multiples of 5 in the array must be included in the group. If the value immediately following a multiple of 5 is 1, it must not be chosen. (No loops needed.)

Example:
Input: groupSum5(0, [2, 5, 10, 4], 19)
Output: True

Input: groupSum5(0, [2, 5, 10, 4], 17)
Output: True

Input: groupSum5(0, [2, 5, 10, 4], 12)
Output: False

Reference:

1. https://codingbat.com/java/Recursion-1
2. https://codingbat.com/java/Recursion-2