Practice 01 (08-03-2021)

Recursion I (Relatively Easy Problems)

Write recursive function for the following problems: Sample runs are given with each question.

1. Given a non-negative integer, return the count of the occurrences of 7 as a digit, so for example 717 has 2 occurrences of digit 7. 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).

count7(717)	2
count7(7)	1
count7(123)	0
count7(77)	2
count7(7123)	1
count7(771237)	3
count7(771737)	4
count7(47571)	2
count7(777777)	6
count7(70701277)	4
count7(777576197)	5
count7(99999)	0
count7(99799)	1

2. 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.

triangle(0) → 0	0
triangle(1) → 1	1
triangle(2) → 3	3
triangle(3) → 6	6
triangle(4) → 10	10
triangle(5) → 15	15
triangle(6) → 21	21
triangle(7) → 28	28

3. Given an array of integers, 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.

array220([1, 2, 20],3)	TRUE
array220([3, 30],2)	TRUE
array220([3, 3, 30, 4], 4)	TRUE
array220([20, 2, 21, 210],4)	TRUE
array220([1, 2, 3, 4, 5, 6],6)	FALSE
array220([1, 2, 3, 4, 5, 50, 6],7)	TRUE
array220([1, 2, 3, 4, 4, 50, 500, 6],8)	TRUE

4. Given a non-negative integer, 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).

`sumDigits(126)`	9
`sumDigits(49)`	13
`sumDigits(12)`	3
`sumDigits(10)`	1
`sumDigits(1)`	1
`sumDigits(0)`	0
`sumDigits(730)`	10

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

"hello"	h*e*l*l*o
"abc"	a*b*c
"ab"	a*b
"a"	a
""	""
"3.14"	3*.*1*4
"Chocolate"	C*h*o*c*o*l*a*t*e