CS 1301 CRN 28045- 04/04/2022

Problems retrieved from CodingBat

Array-2 > bigDiff

Given an array length 1 or more ints, return the difference between the largest and smallest values in the array.

bigDiff([10, 3, 5, 6]) → 7

bigDiff([7, 2, 10, 9]) → 8

bigDiff([2, 10, 7, 2]) → 8

Array-2 > sum13

Return the sum of the numbers in the array, returning 0 for an empty array. Except the number 13 is very unlucky, so it does not count and numbers that come immediately after a 13 also do not count.

sum13([1, 2, 2, 1]) → 6

sum13([1, 1]) → 2

sum13([1, 2, 2, 1, 13]) → 6

Array-2 > lucky13

Given an array of ints, return true if the array contains no 1's and no 3's.

lucky13([0, 2, 4]) → true

lucky13([1, 2, 3]) → false

lucky13([1, 2, 4]) → false

Array-2 > sum28

Given an array of ints, return true if the sum of all the 2's in the array is exactly 8.

sum28([2, 3, 2, 2, 4, 2]) → true

sum28([2, 3, 2, 2, 4, 2, 2]) → false

sum28([1, 2, 3, 4]) → false

Array-2 > more14

Given an array of ints, return true if the number of 1's is greater than the number of 4's

more14([1, 4, 1]) → true

more14([1, 4, 1, 4]) → false

more14([1, 1]) → true

Recursion-1 > array6

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.

array6([1, 6, 4], 0) → true

array6([1, 4], 0) → false

array6([6], 0) → true

Recursion-1 > array11

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.

array11([1, 2, 11], 0) → 1

array11([11, 11], 0) → 2

array11([1, 2, 3, 4], 0) → 0

Recursion-1 > array220

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.

array220([1, 2, 20], 0) → true

array220([3, 30], 0) → true

array220([3], 0) → false

String-2 > sameStarChar

Returns true if for every '*' (star) in the string, if there are chars both immediately before and after the star, they are the same.

sameStarChar("xy*yzz") → true

sameStarChar("xy*zzz") → false

sameStarChar("*xa*az") → true

Array-2 > centeredAverage

Return the "centered" average of an array of ints, which we'll say is the mean average of the values, except ignoring the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. Use int division to produce the final average. You may assume that the array is length 3 or more.

centeredAverage([1, 2, 3, 4, 100]) → 3

centeredAverage([1, 1, 5, 5, 10, 8, 7]) → 5

centeredAverage([-10, -4, -2, -4, -2, 0]) → -3