Duncan Di Mauro

SID: 805163177

February 17, 2021

Report 4

**Obstacles:**

This project was difficult in the beginning but became easier as I finished my first few functions. The main obstacles that I needed to overcome included figuring out that I needed to use a nested loop in order to make many of my functions work. The first loop would go through each element of the input array and the second loop would either go through each character of that element or go through the array again, comparing that specific element to the other elements of the input array. This approach was used in every single one of my functions. Other than this obstacle, the most challenging thing was creating the identicalValuesTogether function. I had trouble thinking of a way to check if identical values were next to each other, but eventually came up with a solution. In my nested loop, I compared one element of the array to each of the other elements and checked if any identical values were found. I then found the difference in indices of both those elements to see how far away they are from each other. Then, I checked to see if all indices going from one element to the other one are identical. If they are, then the function returns true, and false otherwise.

**List of Test Data:**

locateMaximum(const string array[], int n) test data:

([a, b, c, d, e, f, g], 0) --- tests for when n = 0

([a, b, c, d, e, f, g], -3) --- tests for when n < 0

([a, b, c, d, e, f, g], 7) --- tests for a valid array with n = array size

([a, b, c, d, e, f, g], 5) --- tests for a valid array with 0 < n < array size

([g, e, a, g, e, f, g], 7) --- tests for a valid array with duplicate maximum values

([shark, dolphin, whale, elephant, 172, 172, elephant, potato], 7) --- tests for valid array

countFloatingPointValues(const string array[], int n) test data:

([2.7, b, c, d, 5.0, f, 6], 0) --- tests for when n = 0

([2.7, b, c, d, 5.0, f, 6], -1) --- tests for when n < 0

([2.7, b, c, d, 5.0, f, 6], 7) --- tests for a valid array with n = array size

([2.7, b, c, d, 5.0, f, 6], 5) --- tests for a valid array with 0 < n < array size

([a, b, c, d, e, f, g], 7) --- tests for no floating point values

([2.0.2, b, c, d, e, f, g], 7) --- tests for when there are multiple periods in a value

([$2.0, b, c, 3, 0.1, f, 5.9999], 7) --- tests for when there is a non-numerical value or period combined

with a floating point

hasNoCapitals(const string array[], int n) test data:

([a, b, c, d, e, f, g], 0) --- tests for when n = 0

([a, b, c, d, e, f, g], -3) --- tests for when n < 0

([A, b, c, d, e, f, 23], 7) --- tests for a valid array with n = array size, should return false

([2.7, b, c, d, 5.0, f, 6], 5) --- tests for a valid array with 0 < n < array size, should return true

([Zap, fall, winter, d, $5.0, fff, 6], 5) --- tests for an array, should return false

([11, 2, 3, 4, 5.0, 78, 6], 5) --- tests for an array of numbers, should return true

identicalValuesTogether(const string array[], int n) test data:

([a, a, c, c, e, f, g], 0) --- tests for when n = 0

([a, b, c, d, e, f, g], -3) --- tests for when n < 0

([aa, b, aa, b, aa, b, b], 7) --- test should return false

([aa, aa, b, b, b, b], 6) --- test should return true

([%17, elephant, nope, nope, aa, b, b], 7) --- test should return true

([%17, elephant, TURKEY, nope, aa, b, b], 6) --- test should return true

([%17, b, TURKEY, nope, aa, b, a], 6) --- test should return false

hasTwoOrMoreDuplicates(const string array[], int n) test data:

([a, a, c, c, e, f, g], 0) --- tests for when n = 0

([a, b, c, d, e, f, g], -3) --- tests for when n < 0

([abc, abc, c, e, c, f, g], 7) --- test should return true, two duplicates

([abc, abc, c, c, e, g, g], 7) --- test should return true, three duplicates

([ab, ab, c, d, e, g, g], 6) --- test should return false, one duplicate

([a, a, c, d, e, f, g], 0) --- test should return false, one duplicate

([ahj, ahj, ahj, d, e, f, g], 0) --- test should return true, one triplet

shiftLeft(string array[], int n, int amount, string placeholder) test data:

([green, yellow, c, c, e, blue, g], 0, 5, “water”) --- tests for when n = 0

([a, b, c, d, e, f, g], -3, 7, “water”) --- tests for when n < 0

([A, b, c, d , e, f, 23], 7, 5, “lava”) --- tests with n = array size

([A, b, c, d, e, f, 23], 7, 7, “lava”) --- tests with amount = n = array size

([A, b, c, d, e, f, 23], 4, 20, “lava”) --- tests with amount > n < array size

([A, b, c, d, e, f, 23], 3, 3, “lava”) --- tests with amount = n < array size

replaceFirstAndLastOccurences(string array[], int n, char charToFind, char charToReplace) test data:

([green, yellow, a, c, a, blue, g], 0, ‘a’, ‘$’) --- tests for when n = 0

([green, yellow, a, c, a, blue, g], 0, ‘a’, ‘$’) --- tests for when n < 0

([green, yellow, c, c, e, blue, a], 6, ‘a’, ‘$’) --- tests when no characters need replacing

([aabbaac, a 1 2 3 4 a , apple, potato, e, blue, g], 0, ‘a’, ‘$’) --- tests for strings that have more than two of charToFind, only one charToFind, and zero of charToFind

([mississippi, a 1 2 3 4 a , apples, potato, seashore, s, gas], 0, ‘s’, ‘p’) --- same as previous test, and also tests for strings that have charToFind and charToReplace within the same element