CS-546B Lab 2 Comments – Naveen Renji (20016323)

first submission was having console.log() instead of returned output from functions

Naveen Mathews Renji, Sep 22 at 4:50pm

-1; arrayStats does not return object with proper values for valid input Test Case: arrayStats([7, 9, 11, 15, 19, 20, 35, 0]) Expected: { mean: 14.5, median: 13, mode: 0, range: 35, minimum: 0, maximum: 35, count: 8, sum: 116 } |Actual: {"sum":116,"count":8,"mean":14.5,"median":13,"mode":0,"min":0,"max":35,"range":35}

-1; arrayStats does not return object with proper values for valid input Test Case: arrayStats([1, 1, 2.5, 5, 10, 10, 15, 15, 20, 20]) Expected: { mean: 9.95, median: 10, mode: [1, 10, 15, 20], range: 19, minimum: 1, maximum: 20, count: 10, sum: 99.5 }|Actual: {"sum":99.5,"count":10,"mean":9.95,"median":10,"mode":[1,10,15,20],"min":1,"max":20,"range":19}

**For the above two deductions – My output values are correct, but I understand that the object keys in my output are within quotes. but can that please be considered since the function output gives the correct values?**

-3.75; commonElements does not properly return an empty array for no common elements Test Case: commonElements([1, 0, 16.001, ['foo', 'bar'], 7, 7], ['7', false, true, 16.01, ['bar', 'foo']]) Expected: [] |Actual: [["foo","bar"]]

**In the above deduction, since the function checks for common elements of the input arrays, ['bar', 'foo'] and ["foo","bar"] also represents an array within an array that have common elements. So that’s why, I have that as my output… and I hope this can be considered for points.**

-3.75; palindromes does not properly return an array with all the palindromes Test Case: palindromes('The racecar went to a big track. 'Wow!' i, said. zzbbzz?') Expected: ['racecar', 'a', 'Wow', 'i', 'zzbbzz'] |Actual: ["racecar","a","wow","i","zzbbzz"]

**In the above palindromes function, my output is the same as the expected output. Single and double quotes are the only difference. Please consider this**

-3.75; replaceChar does not properly replaced string Test Case: replaceChar('H ') Expected: 'H' |Actual: H\* $

-1; charSwap does not throw for string with less than 4 chars Test Case: charSwap('hello', 'w ')

**I tried this test case, and it threw an error for me just as it should. Could you please re-verify this one? Below is my implementation code and output.**

Text

Description automatically generated

Text

Description automatically generated

-1; charSwap does not properly swap string Test Case: charSwap(' four ', ' lett') Expected: 'lett four' |Actual: lour fett

**Though the above failed for me when I tried it now because I didn’t assign the trimmed string back to the variable, the output I got was different from the above one. Please find below what I got. This just got me wondering why the output was different for me and my grader...**

Text

Description automatically generated with medium confidence

-1.5; deepEquality does not throw for invalid types Test Case: deepEquality({a: 2, b: 4}, [3,4,5]))

**The above test case is syntactically incorrect (because of the parenthesis), so my compiler (VSC) is not even able to compile the code to run it. Could you please re-check this one?**

-3.75; deepEquality does not return true Test Case: deepEquality({b:2, a:2, c:{d:[1,2,3, {z:true, y:false}], e:NaN, f:{g: undefined, h:{}}}}, {a:2, c:{f:{h:{}, g: undefined}, e:NaN, d:[1,2,3, {z: true, y:false}]}, b:2}) Expected: true |Actual: false

**I ran the above test case using the debugger and I observed that it failed only because of “e:NaN”. As NaN != NaN in JavaScript, I understand we must consider the NaN edge case and check for a variable being NaN and have a validation and equality test for NaN as well always, right?**

-3.75; commonKeyValues does not return all common keys Test Case: commonKeysValues({name: {first: "Patrick", last: "Hill", goodbye: [6,4,5]}, age: 46, hello: NaN}, {school: "Stevens", name: {first: "Patrick", last: "Hill", goodbye: [6,4,5]}, hello:NaN} Expected: {name: {first: "Patrick", last: "Hill", goodbye: [6,4,5]}, first: "Patrick", last: "Hill", goodbye: [6,4,5], hello:NaN} |Actual: {"0":6,"1":4,"2":5,"name":{"first":"Patrick","last":"Hill","goodbye":[6,4,5]},"first":"Patrick","last":"Hill","goodbye":[6,4,5]}

-1; calculateObject object values does not throw for non-numbers Test Case: calculateObject({a:NaN, b:true, c:'7'}, n =>n\*2)

Kevin Ha, Sep 28 at 2:05am

+3.75; replaceChar does properly replaced string Test Case: replaceChar('H ') Expected: 'H\* $' |Actual: H\* $

Ryan Gallagher, Sep 30 at 12:07am