CS 31 Project 4 Report

**A) A Brief Description of Notable Obstacles**

In my opinion, this project was easier than project 3, and I did not encounter too many notable obstacles. At first the project seemed overwhelming due to the sheer number of functions, but as I worked through each one I discovered that implementation was actually not that hard. The functions I spent the most time thinking about were flip, subsequence, and separate; however, after some careful thought it was rather straightforward to work out their implementation. One thing to note is that I was unable to complete the bonus challenge for separate where the function does not create an additional array and still works correctly, so I suppose that could be considered an obstacle.

However, one major obstacle was deciding on how each function would handle the 0 case. This was very confusing for me and it was difficult for me to even visualize what an empty array was and what its implications were. I addressed this problem by going to office hours, asking plenty of questions, and ultimately just trying to logically think about how each function should handle a 0 case, and coding it thus. I believe I have correctly addressed the 0 case issue for all functions, but I would not be surprised if I ended up getting it wrong for a couple of function. Other than that there were no other notable obstacles.

**B) Test Data (67 in Total)**

*appendToAll*

Array size is 0 *(someArray, 0, “!!!”)*

Array size is negative *(someArray, -2, “!!!”)*

Array size is positive *(someArray, 2, “!!!”)*

Indicated array size is smaller than actual array *(someArray, 1, “!!!”)*

*lookup*

Array size is 0 *(someArray, 0, “CS”)*

Array size is negative *(someArray, -2, “CS”)*

Array size is positive *(someArray, 2, “CS”)*

Indicated array size is smaller than actual array *(someArray, 1, “CS”)*

Array does not contain target *(someArray, 2, “blahblahblah”)*

Array contains two instances of target *(someArray, 4, “target”)*

*positionOfMax*

Array size is 0 *(someArray, 0)*

Array size is negative *(someArray, -2)*

Array size is positive *(someArray, 2)*

Indicated array size is smaller than actual array *(someArray, 1)*

Array has two strings that are >= to all other strings *(someArray, 5)*

*rotateLeft*

Array size is 0 *(someArray, 0, 1)*

Array size is negative *(someArray, -2, 1)*

Array size is positive *(someArray, 2, 1)*

Indicated array size is smaller than actual array *(someArray, 1, 1)*

Array pos is 0 *(someArray, 2, 0)*

Array pos is negative *(someArray, 2, -1)*

Array pos is greater than or equal to array size *(someArray, 4, 5)*

*countRuns*

Array size is 0 *(someArray, 0)*

Array size is negative *(someArray, -2)*

Array size is positive *(someArray, 2)*

Indicated array size is smaller than actual array *(someArray, 1)*

*flip*

Array size is 0 *(someArray, 0)*

Array size is negative *(someArray, -2)*

Array size is positive *(someArray, 2)*

Indicated array size is smaller than actual array *(someArray, 1)*

*differ*

Both array sizes are 0 *(someArray, 0, anotherArray, 0)*

At least one array size is 0 *(someArray, 3, anotherArray, 0); (someArray, 0, anotherArray, 3)*

Both array sizes are negative *(someArray, -2, anotherArray, -4)*

At least one array size is negative *(someArray, -3, anotherArray, 4); (someArray, 4, anotherArray, -3)*

Both array sizes are positive *(someArray, 4, anotherArray, 4)*

Both indicated array sizes are smaller than actual array *(someArray, 5, anotherArray, 2)*

At least one indicated array size is smaller than actual array *(someArray, 6, anotherArray, 2); (someArray, 2, anotherArray, 6)*

Both arrays are the same up to the point where one runs out *(someArray, 6, anotherArray, 3); (someArray, 3, anotherArray, 6)*

*subsequence*

Both array sizes are 0 *(someArray, 0, anotherArray, 0)*

Array 1 size is 0 *(someArray, 0, anotherArray, 4)*

Array 2 size is 0 *(someArray, 4, anotherArray, 0)*

Both array sizes are negative *(someArray, -2, anotherArray, -4)*

At least one array size is negative *(someArray, -3, anotherArray, 4); (someArray, 4, anotherArray, -3)*

Both array sizes are positive *(someArray, 4, anotherArray, 4)*

Both indicated array sizes are smaller than actual array *(someArray, 5, anotherArray, 2)*

At least one indicated array size is smaller than actual array *(someArray, 6, anotherArray, 2); (someArray, 5, anotherArray, 3)*

Array 2 subsequence does not appear in array 1 *(someArray, 7, anotherArray, 2)*

Array 2 subsequence appears multiple times in array 1 *(someArray, 9, anotherArray, 2)*

*lookupAny*

Both array sizes are 0 *(someArray, 0, anotherArray, 0)*

Only one array size is 0 *(someArray, 3, anotherArray, 0); (someArray, 0, anotherArray, 3)*

Both array sizes are negative *(someArray, -2, anotherArray, -4)*

At least one array size is negative *(someArray, -3, anotherArray, 4); (someArray, 4, anotherArray, -3)*

Both array sizes are positive *(someArray, 4, anotherArray, 4)*

Both indicated array sizes are smaller than actual array *(someArray, 5, anotherArray, 2)*

At least one indicated array size is smaller than actual array *(someArray, 6, anotherArray, 2); (someArray, 5, anotherArray, 3)*

No element of array 1 is equal to any element of array 2 *(someArray, 6, anotherArray, 9)*

Multiple elements of array 1 are equal to an element of array 2 *(someArray, 5, anotherArray, 7)*

Multiple elements of array 2 are equal to an element of array 1 *(someArray, 5, anotherArray, 7)*

*separate*

Array size is 0 *(someArray, 0, “blah”)*

Array size is negative *(someArray, -2, “blah”)*

Array size is positive *(someArray, 2, “blah”)*

Indicated array size is smaller than actual array *(someArray, 1, “blah”)*

Separator is not present within the array *(someArray, 2, “blah”)*

Separator is present within the array *(someArray, 2, “CS”)*

Separator is present multiple times within the array *(someArray, 4, “CS”)*

All elements are less than separator *(someArray, 4, “ZZZZ”)*

Multiple elements are greater than or equal to separator *(someArray, 4, “aa”)*