Ryan Lam

Notable Obstacles:

I had trouble devising an algorithm for the divide function. Figuring out how to separate the words in the array efficiently was difficult. Another obstacle was making sure that my functions did not access outside the index of the arrays. Since c-style arrays do not check if a given index exists, I had to make sure I was not relying on undefined behavior by thoroughly examining my loops and test cases.

Test Cases:

appendToAll({"dianne", "fiona", "ed", "xavier", "greg"}, 5, "!!!"): appendToAll works correctly

appendToAll({"dianne", "fiona", "ed", "xavier", "greg"}, -5, "!!!"): n is negative

appendToAll({}, 5, "!!!"): empty array

appendToAll({"dianne", "fiona", "ed", "xavier", "greg"}, 5, ""): string to append is empty

appendToAll({"dianne", "fiona", "ed", "xavier", "greg"}, 0, "!!!"): n is zero

appendToAll({"dianne", "fiona", "ed", "xavier", "greg"}, 3, "!!!"): n is less than size of array

lookup({ "dianne", "fiona", "ed", "xavier", "greg" }, 5, "dianne"): lookup works correctly

lookup({ "dianne", "fiona", "dianne", "xavier", "greg" }, 5, "dianne"): multiple correct target strings

lookup({ "josh", "fiona", "ed", "xavier", "greg" }, 5, "dianne"): target string not found in array

lookup({ "dianne", "fiona", "ed", "xavier", "greg" }, -5, "dianne"): n is negative

lookup({ "xavier", "fiona", "ed", "dianne", "greg" }, 3, "dianne"): array of section n does not contain target

lookup({ "xavier", "fiona", "ed", "dianne", "greg" }, 0, "dianne"): n is zero

lookup({ "xavier", "fiona", "ed", "dianne", "greg" }, 5, "diAnNE"): target has different case

lookup({}, 5, "dianne"): array is empty

positionOfMax({"dianne", "fiona", "gavin", "xavier", "ed", "betty"}, 6): positionOfMax works correctly

positionOfMax({}, 6): array is empty

positionOfMax({"dianne", "fiona", "gavin", "xavier", "ed", "betty"}, 0): n is zero

positionOfMax({"dianne", "fiona", "gavin", "xavier", "ed", "betty"}, 4): n is less than size of array

positionOfMax({"dianne", "xavier", "gavin", "xavier", "ed", "betty"}, 6): more than one correct string

positionOfMax({"dianne", "fiona", "gavin", "xavier", "ed", "betty"}, -6): n is negative

rotateLeft({"eleni", "dianne", "fiona", "kevin", "gavin"}, 5, 1): rotateLeft works correctly

rotateLeft({"eleni", "dianne", "fiona", "kevin", "gavin"}, 3, 1): n is less than array size

rotateLeft({}, 4, 1): array is empty

rotateLeft({"eleni", "dianne", "fiona", "kevin", "gavin"}, -5, 1): n is negative

countRuns({"xavier", "betty", "john", "john", "ed", "ed", "ed", "john", "john"}, 9): countRuns works correctly

countRuns({"xavier", "betty", "john", "john", "ed", "ed", "ed", "john", "john"}, -9): n is negative

countRuns({}, 9): array is empty

countRuns({"xavier", "betty", "john", "john", "ed", "ed", "ed", "john", "john"}, 0): n is zero

countRuns({"xavier", "betty", "john", "john", "ed", "ed", "ed", "john", "john"}, 5): n is less than array size

flip({"betty", "john", "", "xavier", "kevin", "dianne"}, 6): flip works correctly

flip({"betty", "john", "", "xavier", "kevin", "dianne"}, -6): n is negative

flip({"betty", "john", "", "xavier", "kevin", "dianne"}, 0): n is zero

flip({}, 6): array is empty

flip({"betty", "john", "", "xavier", "kevin", "dianne"}, 3): n is less than array size

differ({"betty", "john", "", "xavier", "kevin", "dianne"}, 6, {"betty", "john", “dianne”}, 3): differ works correctly

differ({"betty", "john", "", "xavier", "kevin", "dianne"}, -6, {"betty", "john", “dianne”}, -3): n is negative

differ({}, 6, {"betty", "john", “dianne”}, 3): empty array

differ({"betty", "john", "", "xavier", "kevin", "dianne" }, 0, { "betty", "john", “dianne” }, 3): n is zero

differ({"betty", "john", "", "xavier", "kevin", "dianne" }, 3, { "betty", "john", “dianne” }, 1): n is less than array size

differ({"betty", "john", "xavier", "kevin", "dianne" }, 5, { "betty", "john", “xavier” }, 3): arrays are equal until run out

differ({"betty", "john", "", "xavier", "kevin", "dianne" }, 6, { "john", “betty”, “dianne” }, 3): arrays begin differently

subsequence({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 7, {"ed", "xavier", "", "eleni"}, 4): subsequence works correctly

subsequence({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, -7, {"ed", "xavier", "", "eleni"}, -4): n is negative

subsequence({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 0, {"ed", "xavier", "", "eleni"}, 4): n is zero

subsequence({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 4, {"ed", "xavier", "", "eleni"}, 2): n is less than array size

subsequence({}, 7, {"ed", "xavier", "", "eleni"}, 4): empty array

subsequence({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 7, {"ed", "eleni", “xavier”}, 3): no contiguous subsequence

subsequence({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 7, {}, 0): subsequence is empty array

subsequence({"greg", "gavin", "ed", "xavier", "", "ed", "xavier"}, 7, {"ed", "xavier", 2): subsequence appears more than once

lookupAny({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 7, {"fiona", "ed", "john"}, 3): lookupAny works correctly

lookupAny({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, -7, {"fiona", "ed", "john"}, -3): n is negative

lookupAny({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 0, {"fiona", "ed", "john"}, 3): n is zero

lookupAny({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 7, {"fred", "brian", "john"}, 3): no equal elements

lookupAny({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 4, {"fiona", "ed", "john"}, 1): n is less than array size

lookupAny({}, 7, {"fiona", "ed", "john"}, 3): empty array

divide({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 7, "fiona"): divide works correctly

divide({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, -7, "fiona"): n is negative

divide({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 0, "fiona"): n is zero

divide({}, 3, "fiona"): empty array

divide({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 7, "fred"): divider is not in array

divide({"greg", "gavin", "ed", "xavier", "", "eleni", "fiona"}, 4, "fiona"): n is less than array size

divide({"ed", "", "eleni", "fiona"}, 4, "fred"): no elements greater than or equal to divider