1. Notable Obstacles

* On split and flip, trying to code without creating an additional array to move the items around. For this I created a string variable that holds the position of one of the items, before it’s being replaced.
* On split, when rearranging the items, what to do with all the strings equal to the separator. For this I sorted in ascending order, therefore the items equal to the separator will naturally stay in the middle.
* The case appendToAll received n = 0. Solved it by returning 0 if n ==0

b) Test Cases

All test under the following arrays:

string list[7] = {“fallacious”, “elephant”, “faintly”, “jack”, “chile”, “apple”, “code”}

string list2[8] = {“fallacious”, “elephant”, “faintly”, “soccer”, “cat”, “smash”, “boulder”, “dog”}

appendToAll Testing

* n is a negative value (appendToAll(list, -1000, “!!!”)); Returns -1
* append a string correctly (appendToAll(list, 5, “:-)”)); Returns n (5) and lista[0] = fallacious:-), …
* n is 0 (appendToAll(list, 0, “kMate”)); Return 0 and no items are changed

lookup Testing

* n is negative (lookup(list, -5, “Julia”)); Returns -1
* item not on the array (lookup(list, 7, “sugar”)); Returns -1, item not on the array
* finds items successfully (lookup(list, 7, “faintly”)); Returns 2
* two elements are equal in the array (\*a[3] = {“ola”, “chao”, “ola”}) (lookup(a, 3, “ola”)); Returns 0, since the lowest match is on 0

positionOfMax testing

* n is negative (positionOfMax(list, -1)); Returns -1
* code runs correctly (positionOfMax(list, 7)); Returns 3 for “Jack”
* list is empty (string a[0]) (positionOfMax(a, 4)); Return -1
* two greatest elements are equal (\*b[4] = {“a”, “z”, “v”, “z”}) (positionOfMax(b, 4)); Returns 1, since both “z” are the greatest, 1 is the smallest index

rotateLeft testing

* n is negative (rotateLeft(list, -1, 1)); Returns -1
* pos is negative (rotateLeft(list, 1, -1)); Returns -1
* code works correctly (rotateLeft(list, 7, 4)); Returns 4. Order becomes: list[4] = “apple”, list[5] = “code”, list[6] = “chile”
* pos is bigger than or equal to n (rotateLeft(list, 4, 100)); Returns -1

countRuns testing

* n is negative (countRuns(list, -2)); Returns -1
* count runs (countRuns(list, 5)); Returns 5
* array is empty (\*b[0];) (countRuns(b, 0)); Returns 0

flip testing

* n is a negative value (flip(list, -1000)); Returns -1
* odd n (flip(list, 3)); Returns 3, list[0] = faintly, list[1] = elephant, list[2] = fallacious
* even n (flip(list, 4)); Returns 4, list[0] = jack, list[1] = faintly, list[2] = elephant, list[3] = fallacious
* n is 0 (flip(list, 0)); Returns 0

differ testing

* n1 is negative (differ(list, -1, list2, 5)); Returns -1
* n2 is negative (differ(list, 2, list2, -4)); Returns -1
* n1 is smaller than n2 (differ(list, 2, list2, 3)); Returns 2
* n2 is smaller than n1 (differ(list, 4, list2, 1)); Returns 1
* reaches max n’s without difference (differ(list, 2, list2, 2)); Returns 2
* different term found (differ,(list, 7, list2, 8)); Returns 3

subsequence testing

* n1 is negative (subsequence(list, -1, list2, 100)); Returns -1
* n2 is negative (subsequence(list, 5, list2, -5)); Returns -1
* subsequence found (subsequence(list, 7, list2, 3)); Returns 0
* no subsequence found (subsequence(list, 7, list2, 6)); Returns -1
* n2 is bigger than n1 (subsequence(list, 2, list2, 8)); Returns -1
* n2 is 0 (subsequence(list, 5, list2, 0)); Returns 0

lookupAny testing

* n1 is negative (lookupAny(list, -2, list2, 5)); Returns -1
* n2 is negative (lookupAny(list, 3, list2, -2)); Returns -1
* no item corresponds to list2 (\*a[2] = {“cricket”, “mom”}) (lookupAny(a, 2, list2, 3)); Returns -1
* items correspond (lookupAny(list, 5, list2, 5)); Returns 0

split testing

* n is negative (split(list, -2, “piazza”)); Returns -1
* all elements less than splitter (split(list, 7, “zimbabwe”)); Returns 7
* all elements bigger than split (split(list, 7, “aang”)); Returns 0
* split is in the array (split(list, 7, “fallacious”)); Returns 5, since list[5] = fallacious
* split is not on the array (split(list, 7, “borussiaDortmund”)); Returns 1, list[0] = apple, list[1] = chile