**Report on Project 4**

a)

Struggled with the three functions: positionOfMax, subsequence and separate.

positionOfMax: how to check if an array is empty.

subsequence: I found how to check whether a2 was a subsequence of a1 but couldn’t quite figure out how to return the value of the starting position.

separate: took a while to figure out the bubble sort method.

b)

appendToAll:

string g[4] = { "jill", "hillary", "gary", "mindy" };

appendToAll(g, 4, "?") - to check all elements of the array

appendToAll(g, 0, "7") - to check no elements of the array

appendToAll(g, 2, "7") - to check only some elements of the array

appendToAll(g, -7, "7") - to check when input is invalid

lookup:

string h[7] = { "jill", "hillary", "donald", "tim", "", "evan", "gary" };

lookup(h, 7, "evan") - to check all the elements of the array

lookup(h, 2, "donald") - to check only some elements of the array

lookup(h, 0, "jill") - to check no elements of the array

lookup(h, -1, "jill") - to check when the input is invalid

lookup(h, 7, "julia") - to check when there is no match

string k[7] = { "jill", "hillary", "donald", "tim", "", "evan", "jill" };

lookup(k, 7, "jill") - to check when there are more than one matching elements

lookup(k, 7, "Jill") - to check if case sensitive

positionOfMax:

string cand[] = { "jill", "hillary", "donald", "tim", "evan", "mike", "tim" };

positionOfMax(cand, 6) - to check if position of max is lowest (“tim” occurs twice)

positionOfMax(cand, 0) - to check no element of the array

positionOfMax(cand, -9) - to check when input is invalid

rotateLeft:

string g[4] = { "jill", "hillary", "gary", "mindy" };

rotateLeft(g, 4, 1) - to check all elements

rotateLeft(g, 4, 0) - to check when pos = 0

rotateLeft(g, 4, -1) - to check when pos < 0

rotateLeft(g, -1, 1) - to check when n = 0

rotateLeft(g, 0, 1) - to check when n < 0

rotateLeft(g, 4, 4) - to check when pos is invalid

rotateLeft(g, 4, 5) - to check when pos is invalid

countRuns:

string d[5] = { "hillary", "hillary", "hillary", "tim", "tim" };

countRuns(d, 5) - to check all elements

countRuns(d, 0) - to check no elements

countRuns(d, -1) - to check when input is invalid

countRuns(d, 3) - to check some elements in the array

flip:

string f[3] = { "gary", "donald", "mike" };

flip(f, 3) - to check all elements

flip(f, 2) - to check some elements

flip(f, 0) - to check no elements

flip(f, -1) - to check when input is invalid

string m[4] = { "gary", "donald", "mike", "julia" };

flip(m, 4) - to check with even number of elements in the string

differ:

string folks[6] = { "ajamu", "mike", "", "tim", "mindy", "bill" };

string group[5] = { "ajamu", "mike", "bill", "", "tim" };

differ(folks, 6, group, 5) - check all elements of both arrays

differ(folks, 2, group, 1) - to check some elements of both array

differ(folks, 0, group, 5) - to check no element of array 1

differ(folks, 6, group, 0) - to check no element of array 2

differ(folks, -7, group, 5) - to check invalid input for array 1

differ(folks, 6, group, -3) - to check invalid input for array 2

differ(folks, 0, group, 0) - to check no element of both arrays

differ(folks, -9, group, -3) - to check invalid input for both arrays

subsequence:

string names[10] = { "evan", "hillary", "mindy", "jill", "ajamu", "gary" };

string names1[10] = { "hillary", "mindy", "jill" };

subsequence(names, 6, names1, 3) == 1) - to check all elements of both arrays

string names2[10] = { "evan", "jill" };

subsequence(names, 5, names2, 2) - to check all elements of both arrays

subsequence(names, 0, names2, 2) - to check no elements of the first array

subsequence(names, 5, names2, 0) - to check no elements of the second array

subsequence(names, 5, names2, 1) - to check only some elements of the second array

subsequence(names, 4, names1, 3) - to check only some elements of the first array

subsequence(names, 0, names1, 0) - to check no elements of both arrays

subsequence(names, -1, names2, 1) - to check invalid input for first array

subsequence(names, 4, names1, -1) - to check invalid input for second array

subsequence(names, -1, names1, -1) - to check invalid input for both arrays

string names3[10] = { "evan", "hillary", "mindy", "jill", "ajamu", "gary" };

string set1[10] = { "bill", "ajamu", "jill", "hillary" };

lookupAny(names3, 6, set1, 4) - to check all elements of both array

string set2[10] = { "tim", "donald" };

lookupAny(names3, 6, set2, 2) - to check all elements of both array

lookupAny(names3, 0, set2, 2) - to check no element of first array

lookupAny(names3, 6, set2, 0) - to check no element of second array

lookupAny(names3, 0, set2, 0) - to check no element of both arrays

lookupAny(names3, 4, set1, 4) - to check some elements of first array

lookupAny(names3, 6, set2, 1) - to check some elements of second array

lookupAny(names3, 3, set1, 3) - to check some elements of both arrays

lookupAny(names3, -1, set1, 4) - to check invalid input for first array

lookupAny(names3, 6, set2, -9) - to check invalid input for second array

lookupAny(names3, -1, set1, -1) - to check invalid input for both arrays

separate:

string cand3[6] = { "donald", "jill", "hillary", "tim", "evan", "bill" };

separate(cand3, 6, "gary") - to check all elements with a separator that is not in the array

string cand4[4] = { "gary", "hillary", "jill", "donald" };

separate(cand4, 4, "hillary") - to check all elements with a separator in the array

separate(cand4, 0, "hillary") - to check no elements

separate(cand4, -1, "yolo") - to check invalid input

separate(cand4, 4, "") - to check for empty string and separator that is less than all elements in the string

separate(cand4, 3, "julia") - to check when separator is > all elements in the string

separate(cand4, 4, " ") - to check when string is just a space character