1. Obstacles
2. Finding a way to count the position at the same time in two array
3. Figuring out how to use the composition of for loop in lookupAny
4. Finding a way to write separate function without creating a new array. I have to approach the function separately from the beginning and from the end in order not to mess the order
5. Test data

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

assert(lookup(h, 7, "evan") == 5); //two "evan" in the array

assert(lookup(h, 7, "donald") == 2); //regular test

assert(lookup(h, 2, "donald") == -1); //no such element

assert(positionOfMax(h, 7) == 3); //normal test

assert(positionOfMax(h, 0) == -1); //no elements in the array

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

assert(differ(h, 4, g, 4) == 2);

assert(appendToAll(g, 4, "?") == 4 && g[0] == "jill?" && g[3] == "mindy?"); //for n = size of array

assert(appendToAll(g, 2, "!") == 2 && g[0] == "jill?!" && g[3] == "mindy?"); //for n < size of array

assert(rotateLeft(g, 4, 1) == 1 && g[1] == "gary?" && g[3] == "hillary?!"); //regular

assert(rotateLeft(g, 4, 3) == 2 && g[0] == "jill?!" && g[3] == "Hillary?!"); //the end

string e[4] = { "donald", "tim", "", "evan" };

assert(subsequence(h, 7, e, 4) == 2); //regular

assert(subsequence(h,7,e,0) == 0); //no element in a2

assert(subsequence(h,3,e,4) == -1); //no such sequence because n1 < n2

assert(subsequence(h,4,e,4) == -1); // no such sequence

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

assert(countRuns(d, 5) == 2); //regular

assert(countRuns(h, 1) == 1); //n < size

string m[4] = {"hillary", "hillary", "hillary", "timm"};

assert(differ(d,5,m,4) == 3); //regular

assert(differ(d,3,m,3) == 3); //both run out

assert(differ(d,4,m,3) == 3); //one runs out

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

assert(lookupAny(h, 7, f, 3) == 2); //regular

assert(lookupAny(h,2,f,3) == -1); //none

assert(flip(f, 3) == 3 && f[0] == "mike" && f[2] == "gary"); //odd number

assert(flip(h, 6) == 6 && h[5] == "jill" && h[0] == "evan" && h[2] == "tim" && h[6] == "evan"); //even number, partial flip

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

assert(separate(cand, 6, "gary") == 3); //regular

string cand2[6] = { "donald", "jill", "hillary", "tim", "evan", "bill" };

assert(separate(cand2, 5, "bill") == 0 && cand2[5] == "bill" && cand2[0] == "donald"); //n < size