1. One of the main problems that I had was trying to figure out in which cases n=0 was okay and for which ones it wasn’t because it felt somewhat unclear from the spec and I felt that in some cases it could be interpreted either way. I think I ended up figuring out for which functions it is a valid input to get n= 0 and which functions it isn’t okay by trying to reason it out and see if it would cause any errors in the code but I am not completely certain for those specific cases.
2. Test cases:

string h[7] = { "melania", "kamala", "donald", "tim", "", "doug", "jd" };

string g[4] = { "melania", "kamala", "jd", "usha" };

//lookup tests

assert(lookup(h, 7, "doug") == 5);//can find string

assert(lookup(h, 7, "donald") == 2);//can find string in smaller array

assert(lookup(h, 2, "donald") == -1);//can't find string

assert(lookup(h, -1, "doug") == -1);//invalid n

assert(lookup(h, 0, "melania") == -1);//0 length array, can't find melania

//position of max tests

assert(positionOfMax(h, 7) == 3);//max value can be found

assert(positionOfMax(h, 0) == -1);//no max value in empty array

assert(positionOfMax(h, -1) == -1);//invalid n

//rotate Left tests

assert(rotateLeft(g, 4, 1) == 1 && g[1] == "jd" && g[3] == "kamala");//rotate left works

assert(rotateLeft(h, 0, 0) == -1);//rotating left nothing doesn't make sense

assert(rotateLeft(g, -1, 4) == -1);//invald n

//count runs tests

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

assert(countRuns(d, 5) == 2);// runs counted correctly (repeated values)

assert(countRuns(h, 7) == 7);// runs counted correctly (unrepeated values)

assert(countRuns(h, 0) == 0);// 0 runs counted in empty array

assert(countRuns(d, -1) == -1);// invalid n

//flip tests

string f[3] = { "jd", "donald", "gwen" };

assert(flip(f, 3) == 3 && f[0] == "gwen" && f[2] == "jd");// flips correctly

assert(flip(f, 0) == 0);// flipping empty array is empty array

assert(flip(f, -1) == -1);// invalid n

//differ tests

string a[7] = { "melania", "kamala", "donald", "tim", "", "doug", "jd" };//ensures the tests work with emptry strings as well

string b[4] = { "melania", "kamala", "jd", "usha" };

assert(differ(a, 4, b, 4) == 2);//differ correctly with same length arrays

assert(differ(a, 7, b, 4) == 2);//differ correctly with different length arrays (first>second)

assert(differ(b, 4, a, 7) == 2);//differ correctly with different length arrays (second>first)

assert(differ(b, 4, b, 4) == 4);//same arrays so no difference

assert(differ(a, 4, b, 0) == 0);//differ correctly with empty array

assert(differ(a, 4, b, -1) == -1);//invalid n

assert(differ(a, -1, b, 4) == -1);//invalid n

//subsequence tests

assert(subsequence(a, 7, b, 2) == 0);//subsequence exists and is found

assert(subsequence(a, 1, b, 2) == -1);//subsequence length is > array length

assert(subsequence(a, 7, f, 3) == -1);//subsequence never found in array

assert(subsequence(a, -1, b, 2) == -1);//invalid n

assert(subsequence(a, 1, b, -2) == -1);//invalid n

assert(subsequence(a, 7, b, 0) == 0);//empty subsequence exists in all sequences

//lookup any tests

string c[3] = { "jd", "donald", "gwen" };

assert(lookupAny(a, 7, c, 3) == 2);//working lookup any (donald found)

assert(lookupAny(a, 7, c, 0) == -1);//no strings in empty array so nothing found

assert(lookupAny(a, 0, c, 3) == -1);//can't find a string in an empty array

assert(lookupAny(a, -1, c, 3) == -1);//invalid n

assert(lookupAny(a, 1, c, -3) == -1);//invalid n

//separate tests

assert(separate(a, 7, "jd") == 3);//working separate with separator in array

assert(separate(a, 7, "je") == 4);//working separate with separator not in array

assert(separate(a, 0, "jd") == 0);//n returned because no elements are less than separator because there are no elements

assert(separate(a, -1, "jd") == -1);//invalid n

assert(separate(a, 7, "zzzz") == 7);//separator is at the end

string q[5] = { "a", "b", "c", "d","e" };

assert(separate(q, 5, "") == 0);//separator is at the beginning

cout << "All tests succeeded" << endl;