Bryan Wong

805111517

Professor Stahl

CS31

February 19, 2019

Project 4 Report

a. During this project, I ran into a few notable obstacles. In the first iteration of my project, I had working code but it would not compile on the g31 linux server because my program involved creating an array with dynamic size. In my original shiftRight( ) function, I would first copy all of the elements from the given array into a new array, which had its size set to whatever size of the original array was inputted. Since this behavior is not allowed, in the case that the inputted size is not valid, g31 did not run. This meant it was back to the drawing board in terms of designing the function. Ultimately, I programmed an algorithm that would shift elements from the back a repeated number of times, then insert the number of placeholders required. Another challenge was finding the right syntax for the character of a string in an array, which ended up being array[index of array].at(index of string).

b. Here is a list of several test cases I used to test my code:

**Initialized Arrays to Test:**

string a[6] = { "alpha", "beta", "gamma", "gamma", "beta", "delta" };

string folks[8] = {

"samwell", "jon", "margaery", "daenerys",

"tyrion", "sansa", "howard123", "jon"

};

string folks2[8] = {

"samwell", "jon", "margaery", "daenerys",

"tyrion", "sansa", "howard123", "jon"

};

string nums[5] = { "1", "24", "eat12", "ligma", "11"};

string letters[5] = { "z", "y", "x", "w"};

**Testing hasDuplicates( ):**

assert(hasDuplicates(a, -1 ) == false); - an array of size -1 should return false

assert(hasDuplicates(a, 0 ) == false); - an array of size 0 should return false

assert(hasDuplicates(a, 1 ) == false); - an array of size 1 should return false

assert(hasDuplicates(a, 2 ) == false); - in array a, first 2 elements are different

assert(hasDuplicates(a, 3 ) == false); - in array a, first 3 elements are different

assert(hasDuplicates(a, 4 ) == true); - in array a, element 3 and element 4 are the same

assert(hasDuplicates(a, 5 ) == true); - in array a, element 3 and element 4 are the same. Should hold true if size is increased.

assert(hasDuplicates(a, 6 ) == true); - in array a, element 3 and element 4 are the same. Should hold true if size is maxed.

assert(hasDuplicates(folks, 8 ) == true); - in array folks, there are duplicates

assert(hasDuplicates(folks, 2 ) == false); - in array folks, the first 2 elements are different

**Testing countAllDigits( ):**

assert(countAllDigits( folks, 8 ) == 3); - there should only be 3 digits in the array folks

assert(countAllDigits( folks, -10 ) == -1); - an array of negative size should return -1

assert(countAllDigits( folks, 0 ) == -1); - an array of size 0 should return -1

assert(countAllDigits( folks, 3 ) == 0); - in first 3 elements of array folks, there should be no digits

assert(countAllDigits( nums, 1 ) == 1); - in first element of array nums, there should be 1 digit

assert(countAllDigits( nums, 2 ) == 3); - in first 2 elements of array nums, there should be 3 digits

assert(countAllDigits( nums, 3 ) == 5); - in first 3 elements of array nums, there should be 5 digits

assert(countAllDigits( nums, 4 ) == 5); - in first 4 elements of array nums, there should be 5 digits

assert(countAllDigits( nums, 5 ) == 7); - in first 5 elements of array nums, there should be 7 digits

**Testing isInDecreasingOrder( ):**

assert(isInDecreasingOrder( folks, -1) == false); - an array of size -1 should return false

assert(isInDecreasingOrder( folks, 0) == true); - an array of size 0 should return true since there are no values to fail condition

assert(isInDecreasingOrder( folks, 1) == true); - an array of size 1 should return true since there are no values to fail condition

assert(isInDecreasingOrder( folks, 2) == true); - array folks has first 2 elements in alphabetically decreasing (reverse alphabetical) order

assert(isInDecreasingOrder( folks, 3) == false); - first 3 elements of array folks is not in decreasing order

assert(isInDecreasingOrder( folks, 8) == false); - 8 elements of array folks is not in decreasing order

assert(isInDecreasingOrder( a, 3) == false); - first 3 elements of array a is not in decreasing order

assert(isInDecreasingOrder( a, 4) == false); - first 4 elements of array a is not in decreasing order

assert(isInDecreasingOrder( letters, 4) == true); - elements of array letters are in decreasing order

**Testing find( ):**

assert(find( folks, 5, "foo" ) == -1); - foo is not found in folks, should return -1

assert(find( folks, 5, "jon" ) == 1); - first instance of jon in folks is at index 1

assert(find( folks, 8, "jon" ) == 1); - first instance of jon in folks is at index 1, even though there is another jon in index 7

assert(find( folks, -1, "jon" ) == -1); - array of size -1 returns -1

assert(find( folks, 0, "jon" ) == -1); - array of size 0 returns -1

assert(find( folks, 5, "daenerys" ) == 3); - first instance of daenerys in folks is at index 3

assert(find( folks, 5, "daene" ) == -1); - “daene”, part of “daenerys” should return -1 since it cannot be found

assert(find( folks, 8, "" ) == -1); - empty string cannot be found in folks

assert(find( folks, 8, "ligma" ) == -1); - ligma cannot be found in folks, return -1

**Testing shiftRight( ):**

assert(shiftRight(folks, 8, 100, "yo gabba gabba") == 8); - putting a placeholder amount greater than size caps out at size

assert(shiftRight(folks, 8, -1, "yo gabba gabba") == -1); - a negative amount returns -1

assert(shiftRight(folks, -1, 5, "yo gabba gabba") == -1); - a negative array size returns -1

assert(shiftRight(folks, 0, 5, "yo gabba gabba") == 0); - a size 0 array results in no shifts

assert(shiftRight(folks, 8, 0, "yo gabba gabba") == 0); - 0 shifts requested results in 0 shifts

assert(shiftRight(folks, 0, 0, "yo gabba gabba") == 0); - 0 shifts and 0 size results in 0 shifts

assert(shiftRight(folks, 8, 1, "yo gabba gabba") == 1); - 1 shift requested results in 1

assert(shiftRight(folks, 8, 2, "yo gabba gabba") == 2); - 2 shifts requested results in 2

assert(shiftRight(folks, 8, 3, "yo gabba gabba") == 3); - 3 shifts requested results in 3

assert(shiftRight(folks, 8, 4, "yo gabba gabba") == 4); - 4 shifts requested results in 4

assert(shiftRight(folks, 8, 8, "yo gabba gabba") == 8); - 8 shifts requested, same as the size, results in 8

assert(shiftRight(folks, 8, 2, "") == 2); - empty string works as placeholder

**Also printed out arrays afterwards to test behavior worked properly.**

**Testing replaceAllCharacters;**

assert(replaceAllCharacters(folks2, 8, ':', 'A') == 0); - no ‘:’s found in array folks2, nothing replaced

assert(replaceAllCharacters(folks2, -1, 'a', '\_') == -1); - size of -1 results in -1

assert(replaceAllCharacters(folks2, -5, 'a', '\_') == -1); - size of -5 results in -1

assert(replaceAllCharacters(folks2, 0, 'a', '\_') == -1); - size of 0 results -1

assert(replaceAllCharacters(folks2, 1, 'a', '\_') == 1); - replacing ‘a’’s in folks2 results in 1

assert(replaceAllCharacters(folks2, 8, 'e', '\_') == 4); - replacing ‘e’s in folks2 results in 4

assert(replaceAllCharacters(folks2, 8, '&', '\_') == 0); - replacing ‘&’s in folks 2 results in 0

**Also printed out arrays afterwards to test behavior worked properly.**