Jordan Golan

UID: 405335328

1. I encountered many obstacles as I was going through this project. I felt very comfortable with all the functions that needed to be used and code involving arrays, .where I experienced problems, however, was in the problem solving. The first problem I experienced was creating code that counted how many times a character repeated. I was originally very confused on how to store a count for each character that was being used. I eventually realized that I could utilize another array to solve my problem. I created an array with 256 inputs, as each one corresponded with a character on the ASCII table. As the code went through each character in a string is counted in the corresponding location in the array. I then printed the value that had the greatest count. This was valid because int values correspond directly with char values. I was able to use this same technique in the first nonrepeated char function. The next obstacle I encountered was in creating the code for the hasReverse function. I first tried to utilize the palindrome code that was provided a couple weeks ago on ccle, but this was not very applicable. I was very lost for a while, attempting many different solutions. One attempt I had was to create a copy of the array and reverse each string in it, then compare. This attempt was not successful, as it required an array with variable input. I eventually worked out reasoning on paper and was able to follow it all the way through on XCode to create code that worked. I utilized many loops that went through the array forwards and backwards and each character in the strings.
2. Copied below is a list of assert functions that I used to test my functions:

Used given arrays people[5], folks[8], and a[6] for tests.

I also created arrays for my tests:

Testpuncutation[4] = {“!!”, “…”, “(.,”};

TestRev[4] = {“A”, “BAS”, “A”, “AHHHHH”};

TestRepeatChar[3] = {“abbb4dddd”, “hello”, “thankyou”}

//locate minimum test

* to test generally if function works

locateMinimum( people, 5 )

* to test when n<=0

locateMinimum(people, 0)

//count punctuation tests

* to test general if function works correctly when there is punctuation

countPunctuation( people, 5 )

* to test If function works correctly when there is no punctuation

countPunctuation( people, 1 )

* to test when n<=0

countPunctuation(people, -1)

* to test when an array is only punctuation

countPunctuation(Testpunctuation,4)

// hasreverse tests

* To test if function works when false

hasReverse( people, 5

* to test when function work when true

hasReverse( folks, 8 )

* to test if function works when n<=0

assert(hasReverse(people, 0)== false);

* to test function when a string in only one char

hasReverse(TestRev, 4)

// highestOccuredCharacters tests

* To test if function works when there is no repeating char

highestOccurredCharacter( people, 5, 0 )

* to test if function works when there is a repeating char

highestOccurredCharacter( people, 5, 2 )

* to test if function works when n<=0

highestOccurredCharacter(folks, 0, 0)

* to test if function works when index > n

highestOccurredCharacter(folks, 3, 5)

* to test if function works when index < 0

highestOccurredCharacter(folks, 3, -1

* to test that the character of lowest value returns when character repeat same amount

highestOccurredCharacter(TestRepeatChar, 3, 2)

// is inceasing order tests

* To test if function works when supposed to return false

isInIncreasingOrder( people, 5 )

* To test if function works when supposed to return true

isInIncreasingOrder( people, 3 )

* to test if function works when n<=0

isInIncreasingOrder(people, -1)

// firstNonRepeatedCharacter tests

* To test if function works with many repeating char

firstNonRepeatedCharacter( people, 5, 1 )

* to test if function works with no repeating char

firstNonRepeatedCharacter( people, 5, 0 )

* to test if function works when n <=0

firstNonRepeatedCharacter(people, -1, -1)

* to test if function works when index > n

firstNonRepeatedCharacter(people, 2, 3)

* to test if function works when index < 0

firstNonRepeatedCharacter(people, 4, -1)

* to test if the function works when all characters are repeated

firstNonRepeatedCharacter(TestRepeatChar, 3, 0)

// is only digits test

* To test if function works when supposed to return false

isOnlyDigits(a, 6 )

* to test if function works when supposed to return true

isOnlyDigits(a, 3 )

* to test if function works when n<=0

isOnlyDigits(a, -1)