Amy Seo

Professor Smallberg

CS 31

7 November 2019

Project 4 Report

1. In the separate function, it was difficult to swap the values that were before and after the separator when there was a value equal to separator within the array. I realized I had to swap the values that are greater than the separator with the next value that is either less than OR equal to the separator. For the subsequence function, I knew how to find if the first element in the array matched, but making sure the consecutive elements were also there was tricky. I realized I had to iterate through the elements in the second array after finding the first element matched and could only return the index after checking all of them and not during the loop.
2. Test cases:

string people[5] = { “donald”, “lindsey”, “fiona”, “rudy”, “mick” };

string p[5] = { “donald”, “lindsey”, “rudy”, “rudy”, “mick” };

string d[6] = { “amy”, “amy”, “viv”, “pat”, “ed”, “ed” };

string f[5] = { “bob”, “chad”, “kyle”, “john” };

string g[6] = { “bob”, “chad”, “kyle”, “john” };

string h[6] = { “bob”, “chad”, “kyle”, “john”, “john”, “john”};

string n1[9] = { “A”, “B”, “C”, “D”, “E”, “F”, “B”, “C”, “F” };

string n2[3] = { “C”, “D”, “E” };

string n3[3] = { “B”, “C”, “F” };

string n4 [6] = {“a”, “A”, “b”, “B”, “c”, “C”};

Works properly: **appendToAll(people, 5, “!!!”);**

Appending string is a string of an integer: **appendToAll (people, 5, “0”);**

N is negative: **appendToAll(people, -2, “!”);**

N is 0: **appendToAll(people, 0, “!”);**

Works properly: **lookup(people, 5, “rudy);**

lower/uppercase sensitivity: l**ookup(people, 5, “RUDy”);**

N is negative: **lookup(people, -2, “rudy”);**

Multiple same strings: **lookup(p, 5, “rudy”);**

Looking up string that is not in array: **lookup(people, 5, “lisa”);**

Works properly: **positionOfMax(people, 5);**

N is negative: **positionOfMax(people, -5);**

N is 0: **positionOfMax(people, 0);**

Multiple same max strings: **positionOfMax(p, 5);**

Upper and lower case strings: **positionOfMax(n4, 6);**

Works properly: **rotateLeft(people, 5, 1);**

N is negative: **rotateLeft(people, -5, 1);**

Pos is negative: **rotateLeft(people, 5, -2);**

Works properly: **countRuns(people, 5);**

Multiple occurences: **countRuns(d, 6);**

N is negative: **countRuns(d, -3);**

N is 0: **countRuns(d, 0);**

Works properly: **flip(people, 3);**

N is negative: **flip(people, -1);**

Works properly: **differ(people, 5, f, 5);**

Same arrays: **differ(f, 5, g, 5);**

Negative values for n: **differ(f, -1, g, -1);**

Works properly: **subsequence(n1, 9, n2, 3);**

First few elements of n2 match some elements in n1, but later in the array, a complete match is found: **subsequence(n1, 9, n3, 3);**

Second array has 0 elements: **subsequence(n1, 0, n3, 0);**

N is negative: **subsequence(n1, -2, n3, -2);**

Works properly: **lookupAny(n1, 9, n2, 3);**

No identical elements: **lookupAny(f, 4, n2, 3);**

N is negative: **lookupAny(n1, 9, n2, -1);**

Works properly: **separate(f, 4, “dad”);**

Separator is in array already: **separate(f, 4, “john”);**

Multiple separators already in array: **separate(h, 6, “john”);**