Brian Tagle

604907076

1. The biggest obstacle I encountered in this project was making sure all my functions returned correct values when values such as 0 were input as the array size. For some of the functions, the function would be unable to do anything if the array was size 0 and had no elements to inspect so it would return -1. On other functions however, the function would still be able to run with an array of size zero and return a value that wasn’t -1. The subsequence function is a good example of a function that can take an array of size 0 and won’t return -1.

2. // tests for appendToAll – making sure !!! is correctly added to the specified elements and also making sure the inputs of 0 and -20 change nothing about the array and return correct values

string people1[5] = { "clark", "peter", "diana", "tony", "selina" };

(people1, 5, "!!!")

string people2[5] = { "clark", "peter", "diana", "tony", "selina" };

(people2, 3, "!!!")

string people3[5] = { "clark", "peter", "diana", "tony", "selina" };

(people3, 0, "!!!")

(people3, -20, "!!!")

//tests for lookup – make sure lookup can correctly find the target in the array of the specified size, also make sure fringe cases like 0 and negative numbers work

string people4[5] = { "clark", "peter", "diana", "tony", "selina" };

(people4, 5, "tony")

(people4, 3, "tony")

(people4, 3, "diana")

(people4, -3, "diana")

(people4, 5, "bob")

(people4, 0, "bob")

//tests for positionOfMax – make sure position of max can find the position of the max (ascii values) and also handles cases like 0 and negative numbers correctly

string hero[6] = { "clark", "peter", "reed", "tony", "diana", "bruce" };

(hero, 6) == 3);

(hero, 1) == 0);

(hero, 3) == 2);

(hero, 0) == -1);

(hero, -1) == -1);

//tests for rotateLeft – make sure rotateLeft correctly rotates the specified element to the end of the array at the SPECIFIED LENGTH and not to the very end unless the full array size is given. Also make sure 0 and negative number inputs for both n and pos work in all combinations

string super[5] = { "logan", "clark", "peter", "sue", "reed" };

(super, 5, 1) == 1 );

(super, 0, 1) == -1);

(super, 0, 0) == -1);

(super, -10, 2) == -1);

(super, 5, -3) == -1);

(super, 3, 0) == 0 )

//tests for CountRuns – Make sure the code looks through the array of specified size and counts the number of unique runs of strings. One problem was not counting the last run but the problem has been fixed. Also check cases when n is 0 or negative

string d[9] = { "tony", "bruce", "steve", "steve", "diana", "diana", "diana", "steve", "steve" };

(countRuns(d, 9)== 5);

(countRuns(d, 8) == 5);

(countRuns(d, 7) == 4);

(countRuns(d, 2) == 2);

(countRuns(d, 0) == 0);

(countRuns(d, -20) == -1);

//tests for flip – flip works correctly and flips the array of specified length around the specified length divided by 2. Also works correctly for 0 and negative numbers

string folks[6] = { "bruce", "steve", "", "tony", "sue", "clark" };

(folks, 6) == 6 )

string folks1[6] = { "bruce", "steve", "", "tony", "sue", "clark" };

(folks1, 4) == 4)

(folks1, -5) == -1 )

(folks1, 0) == 0 )

//tests for differ – looking at two strings and finding the first element in which they differ and if they do not differ return the smallest array size. Also tests for various combinations of 0 and negative numbers for n1 and n2 to make sure the bases are covered.

string folks2[6] = { "bruce", "steve", "", "tony", "sue", "clark" };

string group[5] = { "bruce", "steve", "clark", "", "tony" };

(folks2, 6, group, 5)== 2);

(folks2, 2, group, 1)== 1);

(folks2, -3, group, 1) == -1);

(folks2, 4, group, -7) == -1);

(folks2, -3, group, -50) == -1);

(folks2, 6, group, 1) == 1);

(folks2, 1, group, 5) == 1);

(folks2, 3, group, 3) == 2);

(folks2, 0, group, 0) == 0);

(folks2, 5, group, 0) == 0);

(folks2, 0, group, 5) == 0);

//tests for subsequence – make sure that the function subsequence correctly looks for array a2 in array a1 and returns -1 if it is not there. This subsequence function is annoying because of the fact that arrays of length 0 can work and return 0 in specific cases so when writing the code I must be sure I’m coding correctly accord to the specs. Also make sure as with all previous functions, negative size doesn’t work

string names[10] = { "logan", "reed", "sue", "selina", "bruce", "peter" };

string names1[10] = { "reed", "sue", "selina" };

string names2[10] = { "logan", "selina" };

(names, 6, names1, 3) == 1);

(names, 5, names2, 2) == -1);

(names, 0, names1, 0) == 0);

(names, -5, names1, -20) == -1);

(names2, 2, names2, 2) == 0);

(names2, 1, names2, 2) == -1);

(names2, 2, names2, 1) == 0);

(names, 2, names2, 0) == 0);

(names, 0, names2, 2) == 0);

//tests for lookupAny – make sure the code works correctly by taking an element first from a1 then comparing it to all elements of a2 to see if there is a match anywhere and if there is not take the the next element of a1 and compare again. Also check for fringe cases with combinations of n1 and n2 being 0 and -1 to make sure the code handles them correctly and returns correct values

string names3[10] = { "logan", "reed", "sue", "selina", "bruce", "peter" };

string set1[10] = { "clark", "bruce", "selina", "reed" };

string set2[10] = { "tony", "diana" };

(names3, 6, set1, 4)==1);

(names3, 1, set1, 4) == -1);

(names3, 5, set1, 2) == 4);

(names3, 6, set2, 2)==-1);

(names3, -4, set1, 0) == -1);

(names3, 0, set1, -4) == -1);

(names3, -4, set1, -26) == -1);

(names3, 0, set1, 4) == -1);

(names3, 6, set1, 0) == -1);

(names3, 0, set1, 0) == -1);

//tests for split- make sure that split correctly sorts the array into elements less than the splitter and elements greater than the splitter. According to examples given by the specs and to make my code work in accordance with it, elements greater than splitter will include elements equal to the splitter. Also make sure that 0 and -1 correctly do nothing or return an error respectively.

string hero1[6] = { "clark", "peter", "reed", "tony", "diana", "bruce" };

string hero2[4] = { "reed", "sue", "peter", "steve" };

string hero3[6] = { "clark", "diana", "bruce", "peter", "reed", "tony" };

(hero1, 0, "logan") == 0);

(hero1, -3, "logan") == -1);

(hero1, 6, "logan") == 3);

(hero3, 6, "logan") == 3);

(hero2, 2, "steve") == 1);

string hero4[4] = { "reed", "sue", "peter", "steve" };

(hero4, 4, "steve")==2); // should == 2

(hero1, 1, "logan") == 1);

(hero1, 2, "logan") == 2);

string hero5[6] = { "clark", "peter", "reed", "tony", "diana", "bruce" };

(hero5, 5, "logan") == 2);