Remas Bashanfar

UID: 505597737

CS31

Project 3

Obstacles:

I struggled with the shift-left function as it was hard for me to think of a way to shift things left. However, I was able to finally make it work. I also had a problem with the first and last occurrence function as I did not know how to run it properly when we only have one occurrence as it would overlap every time.

Some Test Cases:

To elaborate on the point about the one occurrence of a character, I had some test cases that were going wrong for a long time but I was able to fix them. An example is

string arrayA25[]= { "AA", "BB", "aa", "bb", "cc", "dd", "ee"};

assert(shiftLeft(arrayA25, 5, 10, "foo")== 5);

For the assertions that my program handled well:

This assertion tests for the maximum string:

string arrayA1[] = {"A","B", "C", "D", "E", "F", "G", "H"};

assert( locateMaximum(arrayA1, 4)==3);

Assertion to make sure that the floats are 12.2, 12.5,3:

string arrayA17[] = {"12.5", "bye", "12.2", "-12.2", "!gosomewhere?", "3"};

assert( countFloatingPointValues(arrayA17, 6)== 3);

Assertion to make sure this is false because not all As are beside each other:

string arrayA9[]= { "A", "A", "B", "C", "A", "C"};

assert(identicalValuesTogether(arrayA9, 6)== **false**);

Assertion to test that 2 identical pairs result in true

string arrayA8[]= { "A", "A", "B", "C", "A", "A"};

assert(hasTwoOrMoreDuplicates(arrayA8, 6)== **true**);

Assertion to make sure all A become H

string arrayA23[]= { "A", "A", "B", "C"};

assert(replaceFirstAndLastOccurrences(arrayA23, 4, 'A', 'H')== 2);

Assertion to test that everything becomes foo in the array

string arrayA22[]= { "A", "A", "B", "C", "A", "A", "A", "A"};

assert(shiftLeft(arrayA22, 8, 8, "foo")== 8);

Here is a list of all assertions I used ( that worked) to check my code:

string arrayA1[] = {"A","B", "C", "D", "E", "F", "G", "H"};

string arrayA2[] = {"hello", "bye", "MAYBE", "321smth", "!gosomewhere?", ""};

string arrayA3[] = {"12.5", "bye", "12.2", "-12.2", "!gosomewhere?", ""};

string arrayA17[] = {"12.5", "bye", "12.2", "-12.2", "!gosomewhere?", "3"};

string arrayA5[] = { "12"};

string arrayA4[] = {"hello", "bye", "maybe", "321smth", "!gosomewhere?", ""};

string arrayA6[] = {"12.5", "----12.4", "12.2", "-12.2", "12...3", "000.5"};

string arrayA18[] = {"12.5", "----12.4", "12.2", "-12.2", "123", "000.5"};

string arrayA7[]= { "A", "A", "B", "C"};

string arrayA23[]= { "A", "A", "B", "C"};

string arrayA8[]= { "A", "A", "B", "C", "A", "A"};

string arrayA9[]= { "A", "A", "B", "C", "A", "C"};

string arrayA10[]= { "A", "B", "C", "D"};

string arrayA11[]= { "a", "a", "b", "a", "a" };

string arrayA12[]= { "A", "A","A", "B", "C", "A", "C"};

string arrayA13[]= { "A", "A", "B", "C", "A", "C"};

string arrayA14[]= { "A", "A", "B", "C", "A", "A", "A", "A"};

string arrayA21[]= { "A", "A", "B", "C", "A", "A", "A", "A"};

string arrayA22[]= { "A", "A", "B", "C", "A", "A", "A", "A"};

string arrayA15[]= { "AAA", "ABA", "ACAAA", "BA"};

string arrayA16[]= { "AAA", "ABA", "ACAAA", "zzzzzA"};

string arrayA20[]= { "AAA", "ABA", "ACAAA", "zzzzzA"};

string arrayA25[]= { "AA", "BB", "aa", "bb", "cc", "dd", "ee"};

assert( locateMaximum(arrayA1, 4)==3);

assert( locateMaximum(arrayA1, 8)==7);

assert( locateMaximum(arrayA2, 6)==0);

assert( locateMaximum(arrayA2, 3)==0);

assert( locateMaximum(arrayA2, 0)==-1);

assert( locateMaximum(arrayA1, 0)==-1);

assert( locateMaximum(arrayA1, 1)==0);

assert(CountDecimals("000.5"));

assert( PureInteger("12222")== **true**);

assert( PureInteger("-12222")== **false**);

assert( PureInteger("r12222")== **false**);

assert( countFloatingPointValues(arrayA5, 1)== 1);

assert( countFloatingPointValues(arrayA6, 2)== 1);

assert( countFloatingPointValues(arrayA1, 8)== 0);

assert( countFloatingPointValues(arrayA17, 6)== 3);

assert( countFloatingPointValues(arrayA18, 6)== 4);//

assert( hasNoCapitals(arrayA1, 3)== **false**);

assert( hasNoCapitals(arrayA4, 6)== **true**);

assert(identicalValuesTogether(arrayA7, 4)== **true**);

assert(identicalValuesTogether(arrayA7, 1)== **true**);

assert(identicalValuesTogether(arrayA7, 0)== **false**);

assert(identicalValuesTogether(arrayA8, 6)== **false**);

assert(identicalValuesTogether(arrayA9, 6)== **false**);

assert(identicalValuesTogether(arrayA10, 4)== **true**);

assert(identicalValuesTogether(arrayA11, 2)== **true**);

assert(identicalValuesTogether(arrayA11, 5)== **false**);

assert(hasTwoOrMoreDuplicates(arrayA11, 5)== **true**);

assert(hasTwoOrMoreDuplicates(arrayA11, 3)== **false**);

assert(hasTwoOrMoreDuplicates(arrayA11, 4)== **true**);

assert(hasTwoOrMoreDuplicates(arrayA11, 1)== **false**);

assert(hasTwoOrMoreDuplicates(arrayA8, 6)== **true**);

assert(hasTwoOrMoreDuplicates(arrayA14, 8)== **true**);

assert(shiftLeft(arrayA14, 8, 0, "foo")== 0);

assert(shiftLeft(arrayA21, 8, 3, "foo")== 3);

assert(shiftLeft(arrayA22, 8, 8, "foo")== 8);

assert(shiftLeft(arrayA10, 4, 4, "foo")== 4);

assert(shiftLeft(arrayA7, 4, 2, "foo")== 2);

assert(shiftLeft(arrayA25, 5, 10, "foo")== 5);

assert(replaceFirstAndLastOccurrences(arrayA23, 4, 'A', 'H')== 2);

assert(replaceFirstAndLastOccurrences(arrayA15, 4, 'A', 'z')== 7);//[ "zAz", "zBz", "zCAAz", "Bz" ]

assert(replaceFirstAndLastOccurrences(arrayA16, 4, 'A', 'z')== 7);//[ "zAz", "zBz", "zCAAz", "zzzzzz" ]

assert(replaceFirstAndLastOccurrences(arrayA20, 4, 'A', 'A')== 7);//[ "AAA", "ABA", "ACAAA", "zzzzzA" ]