Miko Farin (704789591)

CS31 Discussion 1E

Project 4 Report

a. The hardest obstacle I faced coding the project was making the **int countFloatingPointValues( const string array[ ],int n ); function.** This was due to the fact that I had to make sure the parameters made every string a floating point value. This was confusing and I had to make sure that only numerical and one decimal were allowed within the string in an element of an array. Another difficulty I had making the code, was making sure that the nested loops were correct and how to read the string within an element letter by letter for certain functions such as the i**nt replaceAll( string array1[ ], int n, char letterToReplace, char letterToFill ); function.** Another function that gave me difficulty was the removing duplicates function because I made my function group up in the front of the string.

b.Test Data for each Function:

**int locateMinimum(const string array[], int n):**

{

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

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

string people[5] = { "samwell", "jon", "margaery", "daenerys", "tyrion" };

string t[5] = { "beta", "gamma", "gamma", "beta", "delta" };

assert(locateMinimum(a, 0) == -1); // if n <=0 make sure the function returned a -1

assert(locateMinimum(a, 3) == 0); //alpha is minimal string

assert(locateMinimum(b, 3) == 2); // beta is the minimal string in the array

assert(locateMinimum(people, 5) == 3); // daenerys is the minimal string in the array

assert(locateMinimum(b, 6) == 3); // alpha is the minimal string within the whole string

assert(locateMinimum (t, 5) == 1); //beta is the minimal string within the whole string

}

**int countAllPunctuation( const string array[ ], int n );**

{

string data[4] = { "howard-", "ucla.", "howard", "ucla" };

string a[6] = { "a.lpha", "bet?a", "!gamma'", "gam/.;;;ma", "be.?!ta", "!delta-" };

string b[6] = { "delta", "gamma", "beta", "alpha", "beta", "al.pha'" };

assert(countAllPunctuation(data, 4) == 2);

assert(countAllPunctuation(data, 1) == 1);

assert(countAllPunctuation(data, 0) == -1);

assert(countAllPunctuation(a, 3) == 4);

assert(countAllPunctuation(a, 6) == 14);

assert(countAllPunctuation(b, 6) == 2);

\*all test to make sure it counted all punctuation.

}

**int countFloatingPointValues(const string array[], int n)**

{

string data[4] = { "4.4.3.3", "44", "33.098", "33.098a" };

string a[4] = { "1.0", "1", ".0101", "0.0123" };

string b[4] = { "asdf", "1", "19.01", "1.900" };

assert(countFloatingPointValues(data, -1) == -1); // if there are no n or elements

assert(countFloatingPointValues(data, 4) == 2); // to check if the function works

assert(countFloatingPointValues(a, 4) == 4);// to check if the function works

assert(countFloatingPointValues(b, 4) == 3);// to check if the function works

}

**int removeDuplicatedValues(string array[], int n);**

{

string data[4] = { "happy", "days", "happy", "days" };

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

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

string c[5] = { "a", "a", "", "a", "a" };

assert(removeDuplicatedValues(data, 4) == 2); //test the functionality

assert(removeDuplicatedValues(a, 4) == 1); //test the functionality

assert(removeDuplicatedValues(b, 6) == 2); //test the functionality

assert(removeDuplicatedValues(c, 5) == 3); //make sure it does not recount the same string or double count

}

**int replaceAll(string array[], int n, char letterToReplace, char letterToFill);**

{

string data[4] = { "happy", "days", "are here", "again" };

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

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

assert(replaceAll(data, 4, 'e', 'z') == 3); //cycle through every string to make sure it changes the letter

assert(replaceAll(a, 6, 'a', 'z') == 9); // wanted to check if changing the letter would do anything

assert(replaceAll(b, 6, 'd', 'z') == 1); // continue testing

assert(replaceAll(a, 6, 'm', 'z') == 4); // continue testing

}

**int shiftRight( string array[ ], int n, int amount, string placeholder );**

{

string people[5] = { "samwell", "jon", "margaery", "daenerys", "tyrion" };

assert(shiftRight ( people, -5, 3, "foo" )) == -1); // make sure that n <= 0

assert(shiftRight ( people, 5, 3, "foo" ) == 3); // test the function

}