CS31 Lecture 1 Professor Smallberg Leo Gretzinger

Project 5 Report 11/19/18

**Obstacles:**

In this project, I faced many obstacles. The makeProper function proved very difficult to make, while including all of the different premises for a “proper” pattern. The rate function was also very challenging. Perhaps the most challenging aspect of this function was the part where I attempted to split the words of the document into a double array of chars, or an array of C strings. The was very difficult, as there proved to be a lot of undefined behavior and complex algorithm thinking involved.

**Pseudocode:**

makeProper function

check if the first word is valid and if not, send to back

check if second word is valid and if not send to back

check is separator is valid and if not send to back

go through patterns again and remove repeats

validateWord function

checks if the word is more than length 0 and if it contains nothing but letters

makes all letters lowercase if this is true

strRemove function

removes a bad string from an array and shifts the rest of the elements left by looping through

charRemove function

removes a bad character from an array and shifts the rest of the elements left by looping through

intRemove function

removes a bad integer from an array and shifts the rest of the elements left by looping through

patRemove function

combines strRemove and intRemove and removes a bad pattern from an array and shifts the rest of the elements left by looping through

splitWords function

splits the individual words in a C string sentence and places them in a double array of chars (an array of C strings) with a while loop inside a for loop

loop through the double array and remove any strings that just spaces

loop through the double array and remove any characters that are not in the alaphabet

checkBack function

loops backwards from a given point and checks if any elements match a given word

checkForw function

loops forwards from a given point and checks if any elements match a given word

hasPattern function

loops through a double array that contains C strings

checks backwards and forwards for a second word, given the first

checks backwards and forwards for a first word, given the second

rate function

uses splitWords function to split a single character array into a double one

loops through the double array and adds increments one if the document has a pattern, using

the hasPattern function

returns the number of patterns a document contains, as a rating

**Test Cases:**

const int length = 8;

char word1[length][MAX\_WORD\_LENGTH + 1] = { "mad", "deranged", "NEFARIOUS", "half-witted", "robot" , "plot", "have", "NeFaRiOuS"};

char word2[length][MAX\_WORD\_LENGTH + 1] = { "scientist", "robot", "PLOT", "assistant", "deranged", "Nefarious", "mad", "pLoT" };

int separation[length] = { 1, 3, 0, 2, 1, 0, 12, 0};

// This test case thoroughly tests all of the specifications of the makeProper function, including removing any words that half non-alphabetical numbers, changing each pattern so that every letter is lower case, and removing repeated patterns and keeping the pattern with the largest separator

assert(makeProper(word1, word2, separation, length) == 4, word1[0] == “mad”, word1[1] == “deranged”, word1[2] == “nefarious”, word1[3] == “have”, word2[0] == “scientist”, word2[1] == “robot”, word2[2] == “plot”, word2[3] == “mad”, separation[0] == 1, separation[1] == 3, separation[2] == 0, separation[3] == 12);

// Tests if the program correctly handles an empty word

const int length = 1;

char word1[length][MAX\_WORD\_LENGTH + 1] = {“mad”};

char word2[length][MAX\_WORD\_LENGTH + 1] = { };

int separation[length] = {1 };

assert(makeProper(word1, word2, separation, length) == 0);

// Test if the program correctly handles two patterns

const int length = 1;

char word1[length][MAX\_WORD\_LENGTH + 1] = {“mad”, “deranged”};

char word2[length][MAX\_WORD\_LENGTH + 1] = {“scientist”, “robot” };

int separation[length] = {1 ,3};

assert(makeProper(word1, word2, separation, length) == 2);

// Tests if rate function correctly handles multiple spaces between words of patterns

char document[] = "The mad UCLA scientist unleashed a deranged evil giant robot.";

assert(rate(document, word1, word2, separation, mp) == 2);

// Tests if the rate function correctly handles a document with two patterns, and one near the end with spaces

char document2[] = "The mad UCLA scientist unleashed a deranged robot.";

assert(rate(document2, word1, word2, separation, mp)==2);

//Tests if the rate function correctly handles a document with all non-alphabetical characters

char document3[] = "\*\*\*\* 2018 \* \*\*\*";

assert(rate(document3, word1, word2, separation, mp) == 0);

// Tests if the rate functions handles a colon in the middle of the document correctly

char document4[] = " That plot: NEFARIOUS!";

assert(rate(document4, word1, word2, separation, mp) == 1);

// Tests if the rate function marks only one pattern if the pattern repeats

char document5[] = "deranged deranged robot deranged robot robot";

assert(rate(document5, word1, word2, separation, mp) == 1);

// Tests if the rate function handles a dash in the middle of a word correctly

char document6[] = "That scientist said two mad scientists suffer from deranged-robot fever.";

assert(rate(document6, word1, word2, separation, mp) == 0);

// Tests if the rate function handles numbers in the middle of a word correctly

char document6[] = "That scientist said two mad scien345tist suffer from deranged-robot fever.";

assert(rate(document6, word1, word2, separation, mp) == 1);

// Tests if the rate function handles all spaces correctly

char document7[] = “ ”;

assert(rate(document7, word1, word2, separation, mp) == 0);

// Tests if rate function correctly handles multiple spaces between words of patterns and a pattern at the beginning of the document

char document8[] = "Mad scientist mad UCLA scientist unleashed a deranged evil giant robot.";

assert(rate(document8, word1, word2, separation, mp) == 2);

// Tests if the rate function correctly handles words in document over 20 characters

char document9[] = "Mad scientistasldfkjsadfk mad UCLA scientist unleashed a deranged evil giant robot.";

assert(rate(document9, word1, word2, separation, mp) == 2);

All of these tests cases worked for my program in Visual Studio and the g31 program.