Cynthia Nguyen 705767700

CS31 Project 5, jitter.cpp

Notable Obstacles:

1. None of my tests were working because I kept using the same test list for editStandards, without remembering that the editStandards list had changed, I solved by making more test lists for each editStandards I asserted.
2. I had an out of bounds error when I tried to move the first Standard. I realize that I had to break out of my for loops once I moved a standard to the end, or else I’d keep subtracting from int I, until I had negative numbers
3. g31 did not allow the use of “NULL” in replacement of ‘\0’
4. To prevent a standard to count more than one match, I had to use flags and break out of loops

PseudoCode:

int editStandards(array int distance,

array cstring word1,

array cstring word2,

number of standards)

{

return 0 if nStandards is not positive;

loop through all the standards to remove ones with non-positive distance, and non-letter characters;

if the distance is not positive or word1 or word2 does not have at least one character;

move the standard to the end;

decrease nStandards by 1;

skip to next standard, restart main loop;

loop through characters of word1;

if it isn’t a letter, move standard to the end;

decrease nStandards by 1;

skip to next standard, restart main loop;

else, turn the letter lowercase

loop through characters of word2 and do the same standards removal as word1;

for each standard, find another standard which share both a word1 and a word2;

move the standard with the lower distance to the end;

decrease nStandards by 1;

return nStandards;

}

int determinematchlevel ( standards, number of standards, c string – jeet)

{

If there is not a positive number of standards, return matchlevel is 0;

Remove all non-letter characters from the jeet, and make all letters lowercase;

Create an array containing each word in the jeet, separated by spaces;

Loop through the standards;

Find word1 in array of words;

If word1 is found, find word2 within the distance of the standard;

Increase matchlevel;

Move onto next standard if a match is found;

Return matchlevel;

}

Test cases:

int main()

{

//basic test for all lowercase and no standard has both same w1 word and same w2 word

const int Test0 = 7;

int test0dist[Test0] = {

2, 4, 1, 3, 2, 1, 13

};

char test0w1[Test0][MAX\_WORD\_LENGTH+1] = {

"eccentric", "space", "ELECTRIC", "tunnel-boring", "space", "Electric", "were"

};

char test0w2[Test0][MAX\_WORD\_LENGTH+1] = {

"billionare", "capsule", "CAR", "equipment", "capsule", "car", "eccentric"

};

assert(editStandards(test0dist, test0w1, test0w2, Test0) == 4);

printStandards(test0dist, test0w1, test0w2, Test0);

//tests negative distance is not standard form

//testneg1

int testneg1dist[Test0] = { -2, 4, 1, 3, 2, 1 ,13 };

char testneg1w1[Test0][MAX\_WORD\_LENGTH + 1] = {

"eccentric", "space", "ELECTRIC", "tunnel-boring", "space", "Electric", "were"

};

char testneg1w2[Test0][MAX\_WORD\_LENGTH + 1] = {

"billionare", "capsule", "CAR", "equipment", "capsule", "car", "eccentric"

};

assert(editStandards(testneg1dist, testneg1w1, testneg1w2, Test0) == 3);

printStandards(testneg1dist, testneg1w1, testneg1w2, Test0);

//testneg2

int testneg2dist[Test0] = { 2, 4, 1, 3, -2, 1, 13 };

char testneg2w1[Test0][MAX\_WORD\_LENGTH + 1] = {

"eccentric", "space", "ELECTRIC", "tunnel-boring", "space", "Electric", "were"

};

char testneg2w2[Test0][MAX\_WORD\_LENGTH + 1] = {

"billionare", "capsule", "CAR", "equipment", "capsule", "car", "eccentric"

};

assert(editStandards(testneg2dist, testneg2w1, testneg2w2, Test0) == 4);

printStandards(testneg2dist, testneg2w1, testneg2w2, Test0);

//testneg3

int testneg3dist[Test0] = { 2, 4, 1, 3, 2, 1, -13 };

char testneg3w1[Test0][MAX\_WORD\_LENGTH + 1] = {

"eccentric", "space", "ELECTRIC", "tunnel-boring", "space", "Electric", "were"

};

char testneg3w2[Test0][MAX\_WORD\_LENGTH + 1] = {

"billionare", "capsule", "CAR", "equipment", "capsule", "car", "eccentric"

};

assert(editStandards(testneg3dist, testneg3w1, testneg3w2, Test0) == 3);

printStandards(testneg3dist, testneg3w1, testneg3w2, Test0);

//tests for words that are 20char long

// and

//test move first standard to end

const int Test2 = 7;

int test2dist[Test2] = {

2, 4, 1, 3, 2, 1, 13

};

char test2w1[Test2][MAX\_WORD\_LENGTH + 1] = {

"eccentric1", "spaceaaaaaaaaaaaaaa1", "ELECTRIC", "tunnel-boring", "space", "Electricaaaaaaaaaaaa", "were"

};

char test2w2[Test2][MAX\_WORD\_LENGTH + 1] = {

"billionare", "capsule", "CAR", "equipment", "capsule", "car", "eccentric"

};

assert(editStandards(test2dist, test2w1, test2w2, Test2) == 4);// failed, out of boudns error, added flags and break statements to solve

printStandards(test2dist, test2w1, test2w2, Test2);

//tests every word is at least one letter long

const int Test3 = 7;

int test3dist[Test3] = {

2, 4, 1, 3, 2, 1, 13

};

char test3w1[Test3][MAX\_WORD\_LENGTH + 1] = {

"eccentric", "", "ELECTRIC", "tunnel-boring", "space", "Electric", "were"

};

char test3w2[Test3][MAX\_WORD\_LENGTH + 1] = {

"billionare", "capsule", "CAR", "equipment", "capsule", "car", "eccentric"

};

assert(editStandards(test3dist, test3w1, test3w2, Test3) == 4);// failed, did not remove the standard with no character; added if statement to solve

printStandards(test3dist, test3w1, test3w2, Test3);

//tests Convert function to remove all non apha chars

char testA[MAX\_JEET\_LENGTH] = "I'm upset that on Apr. 29th, 2022, my 2 brand-new BMW i7s were stolen!!";

Convert(testA);

assert(strcmp(testA, "im upset that on apr th my brandnew bmw is were stolen") == 0);

const int TEST1\_NSTANDARDS = 4;

int test1dist[TEST1\_NSTANDARDS] = {

2, 4, 1, 13

};

char test1w1[TEST1\_NSTANDARDS][MAX\_WORD\_LENGTH + 1] = {

"eccentric", "space", "electric", "were"

};

char test1w2[TEST1\_NSTANDARDS][MAX\_WORD\_LENGTH + 1] = {

"billionaire", "capsule", "car", "eccentric"

};

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

"The eccentric outspoken billionaire launched a space station cargo capsule.") == 2);//tests basic

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

"The eccentric outspoken billionaire launched a space capsule.") == 2);//tests for multiple spaces

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

"\*\*\*\* 2022 \*\*\*\*") == 0);//tests for no letters

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

" It's an ELECTRIC car!") == 1);//tests for uppercases

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

"space space capsule space capsule capsule") == 1);//test for multiple matches for one standard

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

"Two eccentric billionaires were space-capsule riders.") == 0);//tests words are broken up by spaces

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

" i love space capsule and being eccentric ") == 1); //tests for out of bounds if last word of jeet matches word1, and spaces at the beginning and end

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa") == 0);//tests 280 char

assert(determineMatchLevel(test1dist, test1w1, test1w2, TEST1\_NSTANDARDS,

" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa11111111111111111111111111111111111!!!!!!!!!!!!!!!!!!!!!!! two eccentric billionares were space-capsule riders It's am ELECTRIC car space space capsule space capsule capsule The eccentric outspoken billionare launced a space capsule") == 3);//tests for multiple standards in long jeet

cerr << "All tests succeeded" << endl;