***Project 5 - Due November 20, 2012***

***Obstacles***

While writing this program, I encountered many obstacles. For the normalizeCriteria method, I encountered problems with removing one with identical criteria, and counting how many criteria I had at the end. To remove identical criteria, I created a separate method, similar, to read in the two word arrays and find if any two words were identical. However, before I ran this, I had to make all letters lower case, so that it would find it even if they were input as capitals. Instead of counting how many criteria there were in the end, I decided to start with the original amount of criteria, and every time I removed one, I decremented the count of criteria.

The computeScore brought up several more problems. At first, I attempted to create a local variable to store document, but then was able to just use document. Using document, I placed each word into a two dimensional array that stored a c-string on every line. However, there were multiple problems regarding the format and what was allowed in each word. The most difficult issue was to make sure extra words weren't inserted if there were multiple spaces or included non-alphabetical characters. To fix this, when I encountered a space, I continued incrementing document until I hit an alphabetical character. The other obstacle was figuring out how to only count each criteria once, as I went through each criteria with every possibility. To fix this, I ended up using break to get out of the inner loop, and using a boolean to check if I should get out of the outer loop, and move onto the next criteria.

***Description - Psuedocode***

Method: normalizeCriteria

make all words lower case in word1 and word2

find criteria with same words and keep only greater distance

repeatedly: for all criteria

if distance is less than or equal to 0

remove from valid criteria

repeatedly: for all criteria

if either word is blank

remove from valid criteria

repeatedly: for all criteria

if criteria contains non alphabetical character

remove from valid criteria

return number of remaining criteria

Method: computeScore

Create two-dimensional array to hold words from document

repeatedly: until document reaches '\0'

if: character is alphabetical

place lower case into current location

increase character index of same c-string

if: character is a space

place '\0' in current location

repeatedly: while next character is not alphabetical and not '\0'

advance to next character in document

if: next character does not equal '\0'

set location to zero

increase num to next c-string

repeatedly: for all criteria

repeatedly: for all words in document

if: criteria was matched, break out of loop

if: criteria matches a word in document

if: second criteria matches a word within criteria distance

increment score

break out of loop

return score

***Test Data***

Tested Given Main Routine:

const int TEST1\_NCRITERIA = 4;

int test1dist[TEST1\_NCRITERIA] = {

2, 4, 1, 13

};

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

"mad", "deranged", "nefarious", "have"

};

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

"scientist", "robot", "plot", "mad"

};

assert(computeScore(test1dist, test1w1, test1w2, TEST1\_NCRITERIA,

"The mad UCLA scientist unleashed a deranged evil giant robot.") == 2);

*Tests to see if it works normally*

assert(computeScore(test1dist, test1w1, test1w2, TEST1\_NCRITERIA,

"The mad UCLA scientist unleashed a deranged robot.") == 2);

*Tests multiple spaces*

assert(computeScore(test1dist, test1w1, test1w2, TEST1\_NCRITERIA,

"\*\*\*\* 2012 \*\*\*\*") == 0);

*Tests for non alphabetical characters*

assert(computeScore(test1dist, test1w1, test1w2, TEST1\_NCRITERIA,

" What a NEFARIOUS plot!") == 1);

*Tests uppercase*

assert(computeScore(test1dist, test1w1, test1w2, TEST1\_NCRITERIA,

"deranged deranged robot deranged robot robot") == 1);

*Tests for criteria met multiple times*

assert(computeScore(test1dist, test1w1, test1w2, TEST1\_NCRITERIA,

"Two mad scientists have deranged-robot fever.") == 0);

*Tests second criteria then first criteria*

assert(computeScore(test1dist, test1w1, test1w2, 0,

"Two mad scientists have deranged-robot fever.") == 0);

*Tests for no criteria*

char one[10][MAX\_WORD\_LENGTH + 1] = {"prolific", "kah.us", "SERVER", "lol", "mad", "DERANGED", "mad", "hAd", "Many", ""};

char two[10][MAX\_WORD\_LENGTH + 1] = {"", "hi", "thousands", "l0l", "scientist", "robot", "Scientist", "MaD", "Hello", "any"};

int dis[10] = {1,3,2,5,3,0,1,1,3,2};

int end = 0;

assert(normalizeCriteria(dis,one,two,10) == 4 && strcmp(one[0], "server") == 0);

*Tests normalizeCriteria for all possibilities: Tests non positive distance, empty words, identical match criteria, lower case*

assert(normalizeCriteria(dis,one,two,0) == 0);

*Tests normalizeCriteria for no criteria*