a. A brief description of notable obstacles you overcame.

In this project one of the main obstacles I have was dealing with C-string, because I was so use to the programming method with string and simple logic operation. For many times during the coding process I keep comparing two C-string with == operation instead of the strcmp function and also got some confusion with 2D array for C-sting, but eventually practice makes perfect, I got better with the C-string when writing the rate function relatively when writing the makeProper function. on the other hand, I was also having a bit trouble coming up with a better method to count the match from the document passed into the function, eventually I used a technic learned in previous java class by setting a boolean variable isFound to determine whether or not the loop should keep checking if there was already a match for the corresponding pattern in the document.

Pseudocode for makeProper:

if nPattern is negative return 0

transform word1 and word2 to lower cases

repeatedly for nPattern times tracking with index:

check if word1 and word2 has no characters

check if each element(char)is letter for word1[index]

check if each element(char)is letter for word2[index]

check if the separation is negative

check if there is any repeats in patterns in either order

when more then one pattern appears in word1

when more then one pattern appears in word2

if find any compare their separation

save with the greater separation

if( either of the checks above failed delete this pattern)

rotate word1 left at that index

rotate word2 left at that index

rotate separation left at that index

*decrement number of nPattern*

*decrement index because of the rotation*

After checking each patterns formed by word1, word2 and separation

return the number of patterns

Pseudocode for rate:

if nPattern is negative treated as if it were 0

create C-string to copy document

create a 2D C-string array

repeatedly until document[I]finds ‘\0’:

if document[I] is a letter and length is less then Max word length +1

add to the array

increment word length

else if document[I] is a space and at least a charter stored

it forms a word and should be finished with ‘\0’

increment the number of word

set word length to 0 for the next iteration that form another word

after storing the words formed from the iteration

add’\0’ as the last element of 2D array to be safe

increment the number of the words

create a counter “found” to keep track

repeatedly for nPatterns time :

loop thought each word store in the array

{

found a matched in word1

found a possible match in word2

if the possible match is within the length of the corresponding separation

increment the number of found

mark this pattern as it has been visit and successfully found a match

break the loop

}

already found one match for this pattern

break the loop test and move on to check the next pattern

after each patterns have been check return the result of found

**Test make proper**

test where npatern =0 makeProper(test1, test2, test3, 0)

test when word in word1 is empty makeProper(test1, test2, test3, 2)

test when word in word1 is space

{" ", " deranged", "nefarious"}

test when seperaction is nagative and 0 int sep[TEST1\_NRULES] = {-1, -3, 0};

test when two patterns in the collection have both the same w1 and w2

case1 {"hello", "hello", "aa" }

{"world", "world", "bb" }

{ 1, 1, 3 }

case2 {"hello", "hello", "aa" }

{"world", "world", "bb" }

{ 1, 3, 3 }

case3 {"hello", "world", "aa" }

{"world", "hello", "bb" }

{ 1, 3, 3 }

case4 {"hello", "world", "aa" }

{"world", "hello", "bb" }

{ 1, 1, 3 }

test when word is not letter and space

{"scientist@3", " deranged", "nefarious"}

{"mad", "robot", "plot"}

{5, 3, 0}

test upper cases

{"scientist@3", " AAbBBc", "nefarious"}

{"mad", "robot", "plot"}

{5, 3, 0}

test word length boundary

{"aaaaaaaaaaaaaaaaaaaa", " AAbBBc", "nefarious"}

{"mad", "robot", "plot"}

{5, 3, 0}

**Test rate**

char a[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] =

{"deranged", "AABBcbd", "nefarious"};

char b[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] =

{"robot", "robot", "plot"};

int sep[TEST1\_NRULES] =

{3, 3, 0};

test if the pater is found more than once

rate("deranged robot deranged deranged robot robot" ,a, b, sep, 1)

test when document is null

rate("" ,a, b, sep, 1)

test when document starts with space

rate(" " ,a, b, sep, 1)

rate(" The mad UCLA scientist unleashed a deranged robot.",test1w1, test1w2, test1dist, TEST1\_NRULES)

test if the function translate the document by defintion

char a[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] =

{"deranged", "AABBcbd", "nefarious"};

char b[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] =

{"robot", "robot", "plot"};

int sep[TEST1\_NRULES] = { 1, 3, 0};

rate("deranged robot" ,a, b, sep, 1)

rate("deranged robot" ,a, b, sep, 1)

rate("deranged robot" ,a, b, sep, 1)

rate("deranged &robot" ,a, b, sep, 1);

test upper case and duplicate

char a[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] =

{"aaa", "bbb", "nefarious"};

char b[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] =

{"bbb", "ccc", "plot"};

int sep[TEST1\_NRULES] = {1, 3, 0};

rate("AAA, BBBb, CCC BBB sss aaa" ,a, b, sep, 2)

translation and long document test so it make sure element than letter should be remove

char a[TEST1\_NRULES][MAX\_WORD\_LENGTH+1]

= {"as", "hes", "aaaaaaaaaaaaaaaaaaaa"};

char b[TEST1\_NRULES][MAX\_WORD\_LENGTH+1]

= {"hes", "friends", "as"};

int sep[TEST1\_NRULES]

= {1, -1, 0};

rate("aaaaaaaaaaaaaaaaaaaa a's he's my friend's friend" ,a, b, sep, makeProper(a, b, sep, 3))

char a[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] = {"nov", "im", "that"};

char b[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] = {"th", "mis", "brandnew"};

int sep[TEST1\_NRULES] = {1, 10, 8};

rate("I'm upset that on Nov. 15th, 2018, my 2 brand-new BMW M850is were stolen!!" ,a, b, sep, makeProper(a, b, sep, 3))