Project 5 Report

Siddarth Chalasani

# Notable Obstacles

When I first implemented the makeProper function, it kept going into an infinite loop because I kept checking the last pattern over and over again. I fixed this by replacing ‘i < nPatterns’ with ‘i < count’ so that it stops re-checking patterns that were known to be invalid. Furthermore, I accidentally typed ‘tolower(word1[i][l])’ instead of ‘word1[i][l] = tolower(word1[i][l])’, so I never actually converted the words in the pattern into lowercase, so my code would not work as intended.

When I implemented the rate function, the code was getting too messy and I kept losing track when I tried to trace through it, so I introduced a new sub-function called findNextWord to make my code cleaner and easier to read.

# Program Design

## makeProper Function

{

If negative number of patterns

Return 0;

Counter;

For every pattern {

If empty word or negative separation

Remove pattern;

If contains non-alphabetical character except ‘ ‘

Remove pattern;

Make lowercase;

For every pattern

Delete duplicate with lower separation;

}

Return counter;

}

## remPattern Function

{

For every pattern after position

Make a copy of pattern to the left

}

## rate Function

{

If negative number of patterns

Return 0;

Counter;

cstring document2;

Copy all letters and spaces of document into document 2;

Convert document2 to lowercase;

For every pattern

While pattern isn’t found yet

If document2 has either word1 or word2

If next separation+1 words have the other word in pattern {

Counter++;

Break;

}

Return counter;

}

## returnNextWord Function

{

Ignore whitespace before first word;

Copy next word into temp;

Return position of character after word;

}

# Test Cases

## Test Case Provided by Spec

char w1[8][MAX\_WORD\_LENGTH+1] = {"mad", "deranged", "NEFARIOUS", "half-witted", "robot", "plot", "have", "NeFaRiOuS"};

char w2[8][MAX\_WORD\_LENGTH+1] = {"scientist", "robot", "PLOT", "assistant", "deranged", "Nefarious", "mad", "pLoT"};

int sep[8] = {1, 3, 0, 2, 1, 0, 12, 0};

cout << makeProper(w1, w2, sep, 8) << endl;

for (int i = 0; i < 8; i++) {

for (int j = 0; w1[i][j] != '\0'; j++)

cout << w1[i][j];

cout << endl;

for (int k = 0; w2[i][k] != '\0'; k++)

cout << w2[i][k];

cout << endl;

for (int l = 0; l < 8; l++)

cout << sep[l];

cout << endl;

}

const int TEST1\_NRULES = 4;

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

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

};

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

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

};

int test1dist[TEST1\_NRULES] = {

1, 3, 0, 12

};

assert(rate("The mad UCLA scientist unleashed a deranged evil giant robot.",

test1w1, test1w2, test1dist, TEST1\_NRULES) == 2);

assert(rate("The mad UCLA scientist unleashed a deranged robot.",

test1w1, test1w2, test1dist, TEST1\_NRULES) == 2);

assert(rate("\*\*\*\* 2018 \*\*\*\*",

test1w1, test1w2, test1dist, TEST1\_NRULES) == 0);

assert(rate(" That plot: NEFARIOUS!",

test1w1, test1w2, test1dist, TEST1\_NRULES) == 1);

assert(rate("deranged deranged robot deranged robot robot",

test1w1, test1w2, test1dist, TEST1\_NRULES) == 1);

assert(rate("That scientist said two mad scientists suffer from deranged-robot fever.",

test1w1, test1w2, test1dist, TEST1\_NRULES) == 0);

## General Testing

const int TEST2\_NRULES = 7;

char test2w1[TEST2\_NRULES][MAX\_WORD\_LENGTH + 1] =

{

"bob", "cow", "MatH", "biGBerg", "bIgbErg", "KyLE", "FAILURE"

};

char test2w2[TEST2\_NRULES][MAX\_WORD\_LENGTH + 1] =

{

"core", "moo", "calculus", "David", "dAviD","EdwaRd" , "SUCKS"

};

int test2dist[TEST2\_NRULES] = {

2, 3, 5, 1, 3, 4, -1

};

const int TEST3\_NRULES = 8;

char test3w1[TEST3\_NRULES][MAX\_WORD\_LENGTH + 1] =

{

"door", "dean", "Math32A", "bergBird", "egged", "KyLE", "UCLA", "importance"

};

char test3w2[TEST3\_NRULES][MAX\_WORD\_LENGTH + 1] =

{

"opened", "burger", "calc", "David", "PAuL","EdwaRd" , "Hard", "sleep"

};

int test3dist[TEST3\_NRULES] = {

2, 3, 5, 1, 3, 4, 0, 3

};

const int TEST4\_NRULES = 6;

char test4w1[TEST4\_NRULES][MAX\_WORD\_LENGTH + 1] =

{

"pokemon", "dean", "corbin", "david", "computer","twentycharacterwords"

};

char test4w2[TEST4\_NRULES][MAX\_WORD\_LENGTH + 1] =

{

"pokemon", "burger", "curved", "computer", "daViD", "suck"

};

int test4dist[TEST4\_NRULES] = {

7, 3, 5, 3, 2, 5

};

assert(makeProper(test2w1, test2w2, test2dist, TEST2\_NRULES) == 5 && strcmp(test2w2[4], "edward") ==0);

assert(rate("Bob made a core mistake by letting moo cows do calculus with david bigberg", test2w1, test2w2, test2dist, 5)==2);

assert(rate("K'yle Edward has allowed bob to let the nuclear core meltdown because his math didn't use caLculus", test2w1, test2w2, test2dist, 5)==2);

assert(rate(" a cow will moo when david has done his bigb32@erg math using calculus ", test2w1, test2w2, test2dist, 5)==3);

assert(rate("my dog name'@d DaVID biGbERg likes to mow such as a certain cow named kyLe EDwArd! ", test2w1, test2w2, test2dist, 5)==2);

assert(rate("HALLO @Bigberg, david!!! I heard failing your class sucks- KYLE EDwa!R#D", test2w1, test2w2, test2dist, 5)==2);

assert(makeProper(test3w1, test3w2, test3dist, TEST3\_NRULES) == 7 && strcmp(test3w2[4], "edward") == 0);

assert(rate("at a burger bbq dean paul egged the crowd because bigbiRD told him to at the wedding of david", test3w1, test3w2, test3dist, 7)==2);

assert(rate("The door opened when paul told kyle edward about the importance of sleep", test3w1, test3w2, test3dist, 7)==3);

assert(rate("UcLA is so hard that David Bergbird got all of us a burger34&)' with the d5&8ean", test3w1, test3w2, test3dist, 7)==2);

assert(rate("Bergbird david told us that david had a bergbird", test3w1, test3w2, test3dist, 7)==1);

assert(rate("nothing of importance is here", test3w1, test3w2, test3dist, 7)==0);

assert(makeProper(test4w1, test4w2, test4dist, TEST4\_NRULES) == 5);

assert(rate("The pokemon trainer caught 20 pokemon after corbin curved the exam", test4w1, test4w2, test4dist, 4)==2);

assert(rate("twentycharacterwordses shouldn't suck so much", test4w1, test4w2, test4dist, 4)==0);

assert(rate("the dean ate many burger's before having a 2nd meal", test4w1, test4w2, test4dist, 4)==0);

assert(rate("I took a corbin test and after being curved I got a c- thanks to david and his computer lessons", test4w1, test4w2, test4dist, 4)==2);

assert(rate("\*932041pokemon pokemon corbin pokemon curved pokemon corbin pokemon curved", test4w1, test4w2, test4dist, 4)==2);

## Testing with Empty Words

const int TEST5\_NRULES = 3;

    char test5w1[TEST5\_NRULES][MAX\_WORD\_LENGTH+1] = {

        "", "deranged", ""

    };

    char test5w2[TEST5\_NRULES][MAX\_WORD\_LENGTH+1] = {

        "scientist", "", "plot",

    };

    int test5dist[TEST5\_NRULES] = {

        12, 3, 5

    };

    assert(makeProper(test5w1, test5w2, test5dist, TEST5\_NRULES) == 0);

## Testing with Empty Document

const int TEST6\_NRULES = 4;

char test6w1[TEST6\_NRULES][21] = {

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

};

    char test6w2[TEST6\_NRULES][21] = {

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

    };

    int test6dist[TEST6\_NRULES] = {

        1, 3, 0, 12

    };

assert(rate("", test6w1, test6w2, test6dist, TEST6\_NRULES) == 0);