Sadhana Vadrevu

CS 31

Professor Smallberg

Project 5 Report

**Obstacles:**

One of the biggest issues I ran into during this project was figuring out how to check each word in the document against the words in the pattern array. I wanted to create a 2-D array of C-strings that contained all of the words from the “document” so that they could easily be checked against the words in the pattern array. At first, I was not sure how to separate the document into individual words. Then, I realized that the space character demarcated the end of a word, so I could copy characters from the document into a C-string variable until a space character was reached. If there was a space character, I would add the word to a list of words containing all the words in the document. Once all the words were in an array, I was able to check each one against the pattern words by running through a nested for loop. One other issue I ran into was how to make sure the rate function only counted one match per pattern. I eventually realized that I needed to break out of the inner for loops if the document contained a match to one of the patterns. I did this by including break statements once a match was found.

**Program Description:**

The two major functions in this program are makeProper and rate. MakeProper is a function that goes through 3 arrays that represent a list of patterns, removes all the patterns that are not valid, and returns the number of valid patterns. This function uses a helper function removePattern that removes a pattern at the specified position from the list of patterns. Rate is a function that reads a document and returns the number of patterns that have a match in that document.

***removePattern:*** removes a pattern at a specified position from pattern array

*remove pattern at specified position*

*move each following pattern one index to the left*

*move pattern at specified position to last position in array*

***makeProper:*** removes all invalid patterns from pattern array

*for each pattern,*

*if pattern contains an empty word,*

*remove pattern*

*else if pattern contains words with non-letter characters,*

*remove pattern*

*else if pattern has a negative separation value,*

*remove pattern*

*otherwise,*

*check if there are any repeats of that pattern in pattern array*

*if there is a repeat,*

*remove pattern with smaller separation value*

*return number of patterns left in pattern array*

***rate:*** reads a document and returns number of patterns that have a match

*remove all non-letters and non-spaces from the document*

*create a 2-D array that contains a list of words in the document*

*record number of words in the 2-D array*

*for each pattern,*

*check if word in 2-D array matches a word in the pattern*

*check if any words separation or less positions away from matching word matches the other word in the pattern*

*record match*

*break*

*if a match was found,*

*move on to next pattern*

*return number of matches found*

**Test Cases:**

**makeProper Tests**

const int TEST1\_NRULES = 9;

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

“mad”, “deranged”, “hello”, “NEFARIOUS”, “half-witted”, “robot”, “plot”, “”, “NeFaRiOuS” };

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

“mad”, “robot”, “there” “PLOT”, “as4sistant”, “deranged”, “Nefarious”, “mad”, “pLoT” };

int test1dist[TEST1\_NRULES] = {

1, 3, -7, 0, 2, 6, 4, 12, 0 };

//check that makeProper returns the correct number of patterns

assert (makeProper(test1w1, test1w2, test1dist, TEST1\_NRULES) == 3);

//check that makeProper correctly modifies the array

assert(strcmp(test1w1[0], "mad") == 0);

assert(strcmp(test1w2[1], "deranged") == 0);

assert(test1dist[1] == 6);

assert (makeProper(test1w1, test1w2, test1dist, 0) == 0); //check that makeProper handles 0 patterns correctly

assert (makeProper(test1w1, test1w2, test1dist, -5) == 0); //check that makeProper handles a negative number of patterns correctly

**rate Tests**

const int TEST2\_NRULES = 4;

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

"mad", "deranged", "nefarious", "have" };

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

"scientist", "robot", "plot", "mad" };

int test2dist[TEST2\_NRULES] = {

1, 3, 0, 1 };

assert(rate("The mad UCLA scientist unleashed a deranged evil giant robot.", test2w1, test2w2, test2dist, TEST2\_NRULES) == 2); //works on regular document

assert(rate("The mad UCLA scientist unleashed a deranged evil giant robot.", test2w1, test2w2, test2dist, 0) == 0); //rate handles 0 patterns correctly

assert(rate("The mad UCLA scientist unleashed a deranged evil giant robot.", test2w1, test2w2, test2dist, -3) == 0); //rate handles a negative number of patterns correctly

assert(rate("The mad UCLA scientist unleashed a deranged robot.", test2w1, test2w2, test2dist, TEST2\_NRULES) == 2); //works when document contains multiple spaces

assert(rate("\*\*\*\* 2018 \*\*\*\*", test2w1, test2w2, test2dist, TEST2\_NRULES) == 0); //document contains no letters

assert(rate(" That plot: NEFARIOUS!", test2w1, test2w2, test2dist, TEST2\_NRULES) == 1); //document contains some non-letter characters

assert(rate("deranged deranged robot deranged robot robot", test2w1, test2w2, test2dist, TEST2\_NRULES) == 1); //document contains multiple matches of the same pattern

assert(rate("That scientist said two mad scientists suffer from deranged-robot fever.", test2w1, test2w2, test2dist, TEST2\_NRULES) == 0); //document contains no matches