# Obstacles:

One of the obstacles I had to overcome while writing this program was dealing with duplicate patterns with different separation in the makeProper() function. My solution was to flag duplicate patterns as invalid but to copy and overwrite the separation of the original encounter with the smaller of the separation values found.

Another minor obstacle I overcame was deciding when to increment patternsFound in rate() – this is because of the way for loops in cstrings end with the null terminator. My original problem was that if the very last thing in a string was the word that would make a pattern found, the loop would fail to check if both were found close enough and not flag that pattern as found. I fixed this by changing when in the loop I checked for finding the pattern to right before the end but after the individual word checks

# Description/Pseudocode:

makeProper() works by iterating through each pattern, checking for validity of each individual pattern. It assumes the pattern is valid and then looks for things that would make it invalid. Namely, it searches for either of the words being empty, having a non-alpha character, the separation being negative, or a repeat pattern. If a pattern is found to be repeated, the lower separation value is retained while the second occurrence is removed. Patterns are make to be entirely lowercase if they are valid

*for each pattern given:*

*assume that this pattern is valid*

*if either word is empty:*

*this pattern is invalid*

*if separation is negative:*

*this pattern is invalid*

*if there is a non-letter character:*

*this pattern is invalid*

*if this pattern already exists:*

*this pattern is invalid*

*set the valid pattern’s separation to the lower of the two pattern’s separations*

*if this pattern is valid:*

*make it all lowercase*

*if this pattern is invalid:*

*remove it*

*return starting number of patterns minus bad patterns*

rate() works by first parsing the given document by making it such that words are separated by one space character, upper case letters are made to be were lower case, and all non-alphabetic characters other than spaces are ignored. It proceeds to check each pattern for its occurrence within parsedDoc by setting flags true if a pattern is found and false if it hasn’t been found within *separation* words, thereby making both flags being true indicate the pattern being found. When they are found, rate increments the number of patterns found by 1 and returns the total number of patterns found

*parse the document so it follows requirements*

*for each pattern given:*

*assume neither word has been found*

*if the first word is found in the parsed document:*

*set first word as found*

*if the second word is found in the parsed document:*

*set second word as found*

*if the first word was found more than separation words ago:*

*set first word as not found*

*if the second word was found more than separation words ago:*

*set second word as not found*

*if both are found:*

*increase number of patterns found by one*

*go to next pattern*

*return number of patterns found*

# Test data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| makeProper() test cases | | | | | |
| Case | Reason | word1[] | word2[] | separation[] | nPatterns |
| empty word | To test if the program handles empty strings correctly | [“hello”, “goodbye”, “”, “maybe”] | [“goodbye”, “yes”, “no”, “maybe”] | [1, 3, 2, 4] | 4 |
| separation is negative | To test if the program handles negative separation correctly | [“hello”, “goodbye”, “thereby”, “maybe”] | [“goodbye”, “yes”, “no”, “maybe”] | [1, 3, 2, -1] | 4 |
| non alpha characters | To test if the program handles non alpha characters correctly | [“hello”, “goodbye”, “thereb$”, “maybe”] | [“goodbye”, “yes”, “no”, “maybe”] | [1, 3, 2, 2] | 4 |
| duplicate patterns | To test if the program handles duplicate patterns correctly  Also checking to make sure the lower separation value is kept | [“hello”, “goodbye”, “thereby”, “maybe”, “goodbye”] | [“goodbye”, “yes”, “no”, “maybe”, “yes”] | [1, 3, 2, 2, 1] | 5 |
| uppercase characters | To test if the program makes all uppercase characters lowercase | [“HeLlO”, “gOoDbYe”, “ThErEbY”, “mAyBe”, “GoOdByE”] | [“GoOdByE”, “yEs”, “No”, “MaYbE”, “wHEEEE”] | [2, 3, 6, 1, 4] | 5 |
| multiple duplicate patterns | To make sure the program keeps the minimum separation value with multiple duplicates | [“hello”, “goodbye”, “thereby”, “maybe”, “goodbye”, “yes”] | [“goodbye”, “yes”, “no”, “maybe”, “yes”, “goodbye”] | [1, 3, 2, 2, 5, 1] | 6 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| rate() test cases | | | | | | |
| Case | Reason | document[] | word1[] | word2[] | separation[] | nPatterns |
| non alpha characters in document | To test if the program handles non alpha charaters in document correctly | "The mad UCLA scientist unlea$hed a !deranged evil giant robot.” | [“mad", "deranged", "nefarious", "have"] | ["scientist", "robot", "plot", "mad"] | [1, 3, 0, 12] | 4 |
| multiple spaces between words | To test if the program handles having multiple spaces between words correctly | “The mad UCLA scientist unleashed a deranged evil giant robot.” | [“mad", "deranged", "nefarious", "have"] | ["scientist", "robot", "plot", "mad"] | [1, 3, 0, 12] | 4 |
| document starting with a pattern word | To test if the program handles the document starting with a pattern word correctly | “mad UCLA scientist unleashed a deranged evil giant robot.” | [“mad", "deranged", "nefarious", "have"] | ["scientist", "robot", "plot", "mad"] | [1, 3, 0, 12] | 4 |
| document ending with a pattern word | To test if the program handles the document ending with a pattern word correctly | “The mad UCLA scientist unleashed a deranged evil giant robot” | [“mad", "deranged", "nefarious", "have"] | ["scientist", "robot", "plot", "mad"] | [1, 3, 0, 12] | 4 |
| extra character on pattern word | To test if the program handles having an extra character attached to the pattern word correctly | "That scientist said two mad scientists suffer from deranged robot fever." | [“mad", "deranged", "nefarious", "have"] | ["scientist", "robot", "plot", "mad"] | [1, 3, 0, 12] | 4 |
| special characters between words | To test if the program handles special characters between words correctly (not as a space) | "That scientist said two mad scientists suffer from deranged-robot fever." | [“mad", "deranged", "nefarious", "have"] | ["scientist", "robot", "plot", "mad"] | [1, 3, 0, 12] | 4 |
| zero words between | To test if the program handles zero words between in a pattern correctly | “The mad UCLA scientist unleashed a deranged evil giant robot with a nefarious plot” | [“mad", "deranged", "nefarious", "have"] | ["scientist", "robot", "plot", "mad"] | [1, 3, 0, 12] | 4 |
| word1 is the same as word2 | To test if the program correctly handles having the same word1 as word 2 | “The mad UCLA scientist unleashed a deranged evil giant robot” | [“mad", "deranged", "nefarious", "have"] | ["scientist", "robot", "plot", "have"] | [1, 3, 0, 12] | 4 |