**A brief description of notable obstacles you overcame.**

One obstacle I had was in editStandards in how I go about removing the standard that did not belong in the list of standards. At first, I went about shifting it to the left. However, it was difficult and started to give me massive headaches on where to put the null character and how to correctly shift it to the left so every standard other than the incorrect standard was safely saved in each array. From this sentence “and it doesn't matter what's in elements 4 though 6 of each array” in the spec, I figured it out. I could simply move the incorrect standard to the very end of each array and replace the incorrect standard with the last standard of the array that needs to be observed. After that the for loop would remain in the same index of each array as it would need to see if the replaced arrary (the last standard we replaced it with) is valid. Only if the standard was correct, would it move onto the next index of each array.

**A description of the design of your program. You should use pseudocode in this description where it clarifies the presentation.**

editStandards

Treat a negative nStandards as if it were 0

Initialize a counter for how many standards were deleted

for loop from 0 index to nStandards – how many standards were deleted (leave incrementing empty)

default create a bool and assume that standard is true

The distance is not positive - Need deletion 🡪 bool to false

Word contains no characters - Need Deletion 🡪 bool to false

for loop for word 1 of standard

If it contains a character that is not a letter (for word1) - Need Deletion 🡪 bool to false

Otherwise tolower every character

for loop for word 2 of standard

If it contains a character that is not a letter (for word2) - Need Deletion 🡪 bool to false

Otherwise tolower every character

for loop from 0 to where index of outerloop is

If the standards are equal - Need Deletion 🡪 bool to false, The longer distance replace the shorter distance

if correct became false

Create 3 temp variables (1 for distance, 1 for word1 string, 1 for word2 string)

Save incorrect standard to each temp variable accordingly

Remove incorrect standard by replacing it with the last standard of the array observed

Assign to the last standard of the array the temp variables accordingly

Increment deleted by 1

if correct remains true

Move onto next standard

return nStandards – deleted

determineMatchLevel

Treat a negative nStandards as if it were 0

Create an empty cstring array (translate) to exclude all non-alphabetic characters from the jeet (280 characters)

Initialize counter for translate array

Intialize bool variable wasSpace to false

for loop every character in the jeet

If it is an alphabetic character

wasSpace is false – because it is alphabetic

Save to the current index of translate the lower case of the alphabetic character

Increment counter for translate array by1

Add a null character after the last alphabetic character

else if was a space character

if the previous character was a space (by checking wasSpace is true)– move on to next character of jeet

else the previous character wasn’t a space – wasSpace is set to true and we add a space to the translate array

Increment counter

else it was neither an alphabetic character of a space character

Move on to next character of jeet

Create an empty cstring array (big size) so we can separate each word as elements in an array

🡪 char separate array[1000][ [MAX\_WORD\_LENGTH + 1]

Initialize a separate index counter

Initialize a char index counter

for loop through characters in the translate array

If character is a space of null character

Increment separate index counter by 1 as we need a separation- new element

Reset char index to 0

else (an alphabetic character)

Save it to current element at its char index ie (separate[separateIndex][charIndex])

Increment charIndex by 1

Add a null character after the last alphabetic character

Initialize standards match counter

for loop through each standard

Default bool found – false

Def index where word1 of standard found – 0

Def index where word2 of standard found – 0

for loop through the separate array

If word1 of standard is found in the jeet/separate array

Save index of word1 to word1 index

for loop from where word1 was found to end of separate array

If word2 of standard is found in the jeet/separate array

Save index of word2 to word2 index

If the distance between word 1 and word2 is valid and positive

found 🡪 true

Increment foundStandard by 1

break – to move onto next standard

if found is true break – to move onto next standard

return foundStandard

**A list of the test data that you could use to thoroughly test your program, along with the reason for each test.**

editStandards

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose | Distance | Word1(“” between every word) | Word2 (“” between every word) | nStandards |
| Distance not positive  (To see if it correctly removes the standard and returns correct int value) | {1,2,0,3}  Also  {1,2, -1, 3} | {eccentric, space, electric, were} | {billionaire, capsule, car, happy} | 4 |
| Word1 contains no character  (To see if it correctly removes the standard and returns correct int value) | {1,2,1,3} | {eccentric,, electric, were} | {billionaire, capsule, car, happy} | 4 |
| Word2 contains no character  (To see if it correctly removes the standard and returns correct int value) | {1,2,1,3} | {eccentric,space, electric, were} | {, capsule, car, happy} | 4 |
| Word1 contains a character that is not a letter  (To see if it correctly removes the standard and returns correct int value) | {1,2,1,4,3} | {eccentric, space, electric, tunnel-boring  ,were} | {billionaire, capsule, car, equipment, happy} | 5 |
| Word2  contains a character that is not a letter  (To see if it correctly removes the standard and returns correct int value) | {1,2,1,4,3} | {eccentric, space, electric, tunnelboring  ,were} | {billionaire, capsule, car, equipment!, happy} | 5 |
| Duplicates (larger distance must be kept)  (To see if it correctly removes the standard and returns correct int value) | {1,2,1,3} | {eccentric, space, electric, space} | {billionaire, capsule, car, capsule} | 4 |
| Word1  contains a character that is uppercase  (To see if it correctly lowercases the uppercased word) | {1,2,1,3} | {EcceNTric, space, eleCtrIC, were} | {billionaire, capsule, car, happy} | 4 |
| Word2  contains a character that is uppercase  (To see if it correctly lowercases the uppercased word) | {1,2,1,3} | {eccentric, space, electric, were} | {billionaire, cApsUle, CAr, happy} | 4 |
| All correct edit standards to see if it doesn’t modify it and returns the correct number of standards | {1,2,1,3} | {eccentric, space, electric, were} | {billionaire, capsule, car, happy} | 4 |

determineMatchLevel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Purpose | Distance | Word1 | Word2 | nStandards | jeet |
| To see if it correctly returns the correct match level with jeet with no edit needed (other than period) | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "eccentric"} | 4 | "The eccentric outspoken billionaire launched a space station cargo capsule." |
| To see if it correctly returns the match level with jeet that contains multiple spaces | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "eccentric"} | 4 | "The eccentric outspoken billionaire launched a space capsule." |
| To see if it correctly returns the match level with jeet that starts off with multiple spaces | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "eccentric"} | 4 | “ The eccentric outspoken billionaire launched a space station cargo capsule." |
| To see if it correctly returns the match level with jeet that ends off with multiple spaces | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "eccentric"} | 4 | “The eccentric outspoken billionaire launched a space station cargo capsule . " |
| To see if it correctly returns the match level with jeet that contains nonalphabetic character | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "eccentric"} | 4 | "\*\*\*\* eccentric \* billionaire\*\*" |
| To see if it correctly returns the match level with jeet that contains uppercase letters | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "eccentric"} | 4 | "It's an ELECTRIC car!" |
| To see if it correctly returns the match level with jeet that contains a standard multiple times but should count as just once | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "eccentric"} | 4 | “eccentric billionaire eccentric billionaire eccentric fat billionaire” |
| To see if it correctly returns the match level with jeet that contains a standard that have word1 and word2 the same | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "were"} | 4 | "They were a bunch of rich people at mcdonalds and were eating cheeseburger" |
| To see if it correctly returns the match level with jeet that contains a nonalphabetic character between word1 and word 2 | {2,4,1,13}; | {"eccentric", "space", "electric", "were"} | {"billionaire", "capsule", "car", "eccentric"} | 4 | "The eccentric-billionaire were eating food" |