Obstacles: I ran into numerous obstacles for this project. The first obstacle was trying to print the word and then the correct amount of spaces. Following Smallberg’s later posted advice in the FRQ helped me realize it is easier to read the past than predict the future. So I restarted and focused on printing the spaces after each word. What I learned from this is to be okay with restarting. I spent a lot of time trying to get it to work, but I should have just restarted immediately instead of avoiding it. Another thing I had trouble with was getting the @P@ to work properly. What helped me was to take in @P@ and then instead of trying to predict the future wait for the next word. Similar to the logic from the spaces. I would wait for it to see if there is a next word until I do what @P@ is supposed to do. Lastly, I also made some off-by-one errors. For example, when subtracting to get the updated line length at the start I forgot about spaces. Another off-by-one error is when to stop for my for loop. I added my null byte at k + 1, but since it starts counting at 0 the plus one is not needed.

Pseudocode:

Implement render function taking in character per line, input, and output

Decare needed bools, ints, and c strings

If linelength is negative

Return 2

Repeatedly: take in line by line

Repeatedly take in next word by calling next word function

If its the first word and its equal to @P@   
 Empty the word

Continue

If its not the first word and equal to @P@   
 If not a previous paragraph break

Set previous break to true

Continue

If the length of the first word is greater than linelength

Call greater line length function

Set wordprinted and return test to true

Else if its the start of the line

If the previous was a paragraph break

Output two new lines

Reset line length remaining

Print current word

Set word printed equals true

Update the remaining line length

If the word is not equal to @P@   
 Set previous paragraph break to false

Else if it contains a punctuation

If it fits in the line

Print 2 new lines

Update line length

If its not a previous paragraph break

Output two spaces

Print current word

Set word printed to true

Update remaining characters line

Set contains punctuation to false

If its not a @P@ set prev paragraph break to false

Else

Print word on new line

Set word printed to true

Find the remaining line length

Set contains punctuation to false

If its not a @P@ set prev paragraph break to false

` else if it is a word portion

If the line of teh word is less or equal to the remaining character in the line

If prev was a @P@

Print 2 lines and update remaining line length

Print word

Updated word printed, remaining line length, and previous paragraph break

Else

Print a new line the word

Update remaining character, word pritnted, punctuation, and prev paragraph

Else if

If the current word is greater than line length

Handle if there a prev para break

Print word and update line length

Else if word length is less than remaining

Handle if there a prev para break

Print word and update line length

If punctuation

Set punctuation to true

Return 1 or 0 depending if word was longer than line length

Implement next word function

Get rid of spaces

Repeatedly go through characters till no longer a character

Treat word portions

Implement greater line length function

Reapteadly go through the word and split it into multiple words

Print smaller words

Update the right remaining count

Test cases: for all test cases I run the paragraph through my input render(varies, input, output)

With character per line 5, 10, 15

@P@ test-test @P@

hello test @P@ @P@

testing @P@

This tests numerous things at once, multiple @P@s in a row, @P@ at the start, @P@ at the end, hyphens, and normal words,

With character per line 2, 10, 15

Testing-test hello sir welcome to this omg

Pray for me please hi sad songs – -

Really–line mad comp-sci

This tests multiple hyphens, singular hyphens, hyphens by itself, word is greater than allowed character count per line, checks if it returns the right return value

With character per line -4, 6, 1

Computer-science hello world please be nice

hyphen—-check - @P@ @p@ sad chad mad

check

rad

Checks if the right return value with negative max character line, regular words, word over allowed character count, checks @P@, check fake @p@

With character per line -4, 3, 6

Computer, science. Test-testing:

Hell; heaven() test!

Checks different punctuation, checks regular words, checks to see if it is the right return value (first one should return 2, second and 3rd should be 1)

@P@ Testing–testing hello. Chimp

Champ; testing—

—-- @P@ —- said hello @P@

Please please-plaset.

Hundred testing-test happy test lunch food

yummy @P@

Checks for double punctuation, test @P@ at the start, hyphens, ends with @P@