1. Some of the obstacles I overcame was processing the spaces after punctuation. Initiallys, I put that part where @P@ was processed, but after a few tries of testing I decided to put it somewhere else.
2. Another difficulty I had was implementing the hyphen function. To tackle this, using a whiteboard, I planned out my code making changes there while tracing out some tester inputs on my whiteboard to see if it worded. Then, I created the renderOut function on the whiteboard. Doing Pseodocode on whiteboard helped me efficiently work through this obstacle

PseudoCode

State all #includes and using namespace std

Declaring functions

Int render(int lineLength, istream& inf, ostream& outf){

If length invalid, return 2

Set charCount =0;

intitialize char c to nothing.

Initialize char array with 180 max characters

\*Call all other Boolean function used for future functions (will be used as a reference parameter)

While taking in character{

If not space, put character into array

If space

{

If first word is not “@P@” process otherwise do nothing

If fails, set variable success (initialized at \*) to false

If not first word, process function

If fails, set success to false

}

Processes the last word and if fails, sets success to false

Returns based on success boolean

}

}

void processBreaks(ostream& outf){

output two breaks

}

bool processWord(//Boolean parameters, output destination, and word){

if found @P@ sets boolean variable breaks to true, and until next word is not @P@, won’t process the processBreaks and processRender function and sets check to success of the latter function

if Punctuation found in word set isPunct\* to true

if @P@ not found before, processRender and set check to success of the latter function

}

bool processRender(//Boolean paramters, output destination, and word){

finds size of word and sets into wordSize;

intitializes check and success to determine success of function.

Initializes a word portion array, size max 180

While looping through word{

If normal character present without it being a dash, put it in array with index counter, then check if the character isn’t the last character of th word, if so, process it

If dash present, set null char, process whataver there was in the array and set counter to 0

}

}

Bool render out{

If first portion{

If punctation present, see if two spaces fits within the line and output and set counter +=2. If not endline

If not present, see if a single space first within the line and output accordingly similar to the text above but with one space and counter+=1.

Sets firstPortion to true so this wont come back again until next word present

}

boolean variable to tell the function that it doesn't fit

if word fits within length{

output word and set counter to counter+ wordportion length

} else{

if not already on a new line, produce new line and counter=0

checks if the word fits on the new line, if not set the Boolean variable to false, and wrap text accordingly using a loop iterating through the charatcers of the word portion

}}

Test Cases

Invalid/Valid Input

|  |  |  |
| --- | --- | --- |
| Test Case Number +Reason | Test | Pass? |
| 1. Invalid length  * Checking invalid lengths | testRender(0, "irrelevant", "irrelevant", 2); |  |
| 1. Invalid length  * Checking invalid ength | testRender(-1, "irrelevant", "irrelevant", 2); |  |
| 1. Empty input  * Checking if file remains empty for empty input | testRender(7, "", "", 0);  testRender(251, "", "", 0);  testRender(-7, "", "", 2); |  |
| 1. Tabs and new lines- checking if file remains empty for various lengths | testRender(6, "\n", "", 0);  testRender(6, "\t", "", 0);  testRender(23, "\t\t \t \t", "", 0);  testRender(100, "\t \n\n \n \t\t", "", 0); |  |
|  |  |  |

Length

|  |  |  |
| --- | --- | --- |
| Test Case Number | Test | Passed? |
| 1. Checking if text works with lengths greater enough to wrap around text | testRender(7, "a", "a\n", 0);  testRender(7, "a ab", "a ab\n", 0);  testRender(7, "a ab.", "a ab.\n", 0); | Yes |
| 1. Input text with length 270. Checking input lines with render length 270 (checking with comparison to the render test box given on the SPEC) | It always does seem to me that I am doing more work than  I should do. It is not that I object to the work, mind you;  I like work: it fascinates me. I can sit and look at it for hours.  I love to keep it by me: the idea of getting  rid  of it nearly breaks my heart. @P@ You cannot give me too  much work; to accumulate work has almost become  a passion with me: my study is so full of it now, that there is hardly  an inch of room for any more. | Yes |
| 1. checking if text that cant fit length will be wrapped around correctly even if length is not big enough and returns -1 (checked through the project tester input box)   checking to see if ignoring new line character | Input file :  This is a good day  .  Length 3 | Yes |
| 1. Checking if line can’t fit text again and returns 1 | Input file:  It always does seem to me that I am doing more work than  I should do. @P@It is not that I object to the work  .  Length=4 | 1st test- failed as there was space after period and there was a extra space where there shouldn’t be  2nd test- passed |
| 1. Check if text that is equal to max word length still outputs 0 | File: Hello  5 |  |
| 1. Checking if the function returns 1 again and the spaces behind punctuation are correct   Checking to see if ignoring tab and newline characters | Input file: It always does seem to me that I am doing more work than  I should do. It is not that I object to the work, mind you;  I like work: it fascinates me! I can sit and look at it for hours.  I love to keep it by me: the idea of getting  rid  of it nearly breaks my heart. @P@ You cannot give me too  much work; to accumulate work has almost become  a passion with me: my study is so full of it now, that there is hardly  an inch of room for any more.  Length =8 | 1st trial: the spaces weren’t outputting properly, and thus had to fix some code  2nd trial: worked |

Paragraph & Breaks

|  |  |  |
| --- | --- | --- |
| 1. Tests to see if a single @P@ produces nothing. Also checking if it is case sensitive and takes that particular word only | @P@  Length=4; | Yes |
| 1. Checks whether to see two @P@ produces anything | @P@  @P@  Length=4 | Yes |
| 1. Checks whether word between two paragraph words produces a word and a single endline. @P@ before and after checking if they produce valid breaks | @P@ a @P@  (LENGTH IS 10 for all further testcases in this section) | Yes |
| 1. Checks whether paragraphs breaks are correctly placed between words | @P@ a @P@ b | Yes |
| 1. Checks whether the function correctly produces output with period included (should produce no spaces) | Word. @P@ Word. | Yes |
| 1. Checks if the extra period is considered as a part of a word | Word. @P@ .word. | Yes |
| 1. Checks if @P@ part of a word just produces a word | Word@P@ B |  |

Punctuation and Spaces

|  |  |  |
| --- | --- | --- |
| 1. For these tests in this section, the input file has the same text throughout but checking how it aligns normal spaces and punctuation spaces at different lengths | Input file: It always does seem to me that I am doing more work than  I should do. It is not that I object to the work, mind you;  I like work: it fascinates me! I can sit and look at it for hours.  I love to keep it by me: the idea of getting  rid  of it nearly breaks my heart. @P@ You cannot give me too  much work; to accumulate work has almost become  a passion with me: my study is so full of it now, that there is hardly  an inch of room for any more.  Length =10 | Yes |
|  | Length=3 | Yes |
|  | Length=269 | Yes |
|  | Length=7 | yes |
|  | Length=50 | Yes |

Hyphens

|  |  |  |
| --- | --- | --- |
| 1. This testing section addressed Hyphens and how it correctly divides and spaces text, and whether it correctly returns 1 0r 0. | SAME TEXT FILE USED WITH DIFFERENT LENGTHS  (tests all the different types of hyphens specified in the SPEC):  Thames so-called Henley-but-on-Baily @P@  so--called at---part  fo-  -to  6 | Yes- correctly outputs spacing and returns 1 |
|  | Length=3 | Yes- same as above |
|  | Length=8 | Yes- produces spacing correctly, correctly breaks word portions and produces 0. |
|  | Length=400 | Correctly identifies word portions and breaks them accordingly to length |