I faced a number of obstacles with this project, mainly dealing with dealing with word portions and when word portions were longer than the line length. With word portions, I was not sure how to store the fact that there was a word portion, which I thought I needed to do so that the next word portion could be handled correctly. I solved this problem by printing out the word portion as soon as it was found to be a word portion (ending with ‘-’). However, this did not deal with cases like “so-”, so I had to add a little bit to the if statement that handled word portions to check if the next character was a space or not.

Dealing with word portions larger than the line length involved me rewriting an entire chunk of code. Originally, I handled this case by simply printing out the word stored when it got to the specified line length. However, this made it so that paragraph break would not be read correctly when the specified line length was 1 or 2. In rewriting the code, I used the modulus operator and for loop for the same effect, but one that worked with paragraph breaks. However, I still had to find out how to store if a word portion had been printed, since my solution to the first obstacle I faced did not do that. I just put in another bool variable to do so. I ran into one last problem with this case due to my use of continue. Because I used continue, a character would not be processed, and would instead be skipped. To fix this, I put the code that would normally handle the character before the continue.

The last obstacle I faced I only found out due to the tester program; my program relied on undefined behavior somewhere. After some hunting, I found it was accessing part of an array it should not have. It was easily fixed with some parentheses.

My program first checks for a valid line length; if the line length is invalid, 2 is returned right away. Then it finds a word or word portion and prints out the word or word portion. The word or word portion is then printed out in accordance to the line length. There are two special cases: when the word or word portion is longer than the line length and when the word signals a paragraph break. This is the most important part of the code and is shown in pseudo-code below:

*while a character is found*

*if word or word portion is found*

*if paragraph break signaled*

*if paragraph break not already signaled and no words printed to line*

*store the paragraph break*

*reset word*

*continue*

*store word-found*

*if paragraph-break stored*

*print out paragraph break*

*reset count, paragraph-break storage, and word-printed-to-line storage*

*if word or word portion is too long*

*set return code to 1*

*if anything has been printed to the line*

*print to the next line*

*repeatedly:*

*print out the characters of the word*

*if multiple of line length reached and not at the end*

*print to the next line*

*if it was a word portion*

*reset word-printed-to-line storage*

*store word-portion-printed-to-line*

*if it was a word*

*store word-printed-to-line*

*reset word-portion-printed-to-line storage*

*check and store sentence-end*

*set count to remainder of word length/line length*

*if remainder is 0*

*set count to line length*

*reset word*

*if character is not whitespace*

*add to word*

*continue*

*if word-printed-to-line*

*if sentence-end*

*if word preceded by two spaces cannot fit*

*print to the next line*

*reset count*

*else*

*print two spaces*

*increment count by 2*

*else*

*if word preceded by a space cannot fit*

*print to the next line*

*reset count*

*else*

*print a space*

*increment count*

*if word or word portion cannot fit*

*print to the next line*

*reset count*

*print out word*

*add word length to count*

*if it was word portion*

*reset word-printed-to-line storage*

*store word-portion-printed-to-line*

*if it was a word*

*store word-printed-to-line*

*reset word-portion-printed-to-line storage*

*check and store sentence-end*

*reset word*

*if character is not whitespace*

*add character to word*

At this point, if there is a word left over, the word will be dealt with almost identically to the above. The only differences are that if the word signals a paragraph break, it will be ignored, and that only the portions of code dealing with output will be used. Additionally, if there is no word left over, but words were output, there is an if statement to correctly output the final newline character.

Test Data:

10, “This

is a

test.”

Basic case, no spaces at the end.

10, “This

is a

test. ”

Basic case, spaces at the end.

3, “ab-abc”

One hyphen, no spaces at end.

3, “ab-abc ”

One hyphen, spaces at end.

3, “ab-ab-abc”

Multiple hyphens, no spaces at end.

3, “ab-ab-abc ”

Multiple hyphens, spaces at end.

7, “ab-ab-abc”

Multiple hyphens, only one break.

5, “#P# #P# abc #P# #P# cba #P# #P#”

Paragraph breaks before, in between, and after, no spaces at end.

5, “#P# #P# abc #P# #P# cba #P# #P# ”

Paragraph breaks before, in between, and after, spaces at end.

5, “#P# #P# abc #P# #P# cba #P# #P# def #P# #P#”

Multiple in between paragraph breaks.

6, “”

Nothing in input.

6, “ ”

No words in input.

5, “a. t t.”

Period present, allowing further output on line.

5, “a? t t.”

Question mark present, allowing further output on line.

4, “a. t t.”

Period present, disallowing further output on line.

4, “a? t t.”

Question mark present, disallowing further output on line.

3, “abcde abcd a abcd abc”

Words longer than line length, no spaces at the end.

3, “abcde abcd a abcd abc ”

Words longer than line length, spaces at the end.

3, “abcde abcd-a abcd abc”

Words and word portion longer than line length, no spaces at the end.

3, “abcde abcd-a abcd abc ”

Words and word portion longer than line length, spaces at the end.

3, “abcde #P# abcd-a #P# abcd abc”

Words and word portion longer than line length, with paragraph breaks.

-1, “abcde #P# abcd-a #P# abcd abc”

Invalid line length.

4, “abcde #P# abc-a #P# abcd. a abcde-a”

Words and word portion longer than line length, with paragraph breaks, and a period is present.

4, “abcde #P# abc-abcde #P# abcd? #P# a abcde-a”

Words and word portion longer than line length, with paragraph breaks, and a question mark is present.