1. The most notable obstacle that I encountered was discovering that the strtok function I used for separated words from whitespace in the input could only be used on one C string at a time. Since I used two C Strings to process hyphens and non-hyphenated words, the strtok would not work correctly when a non-hyphenated word needed to be processed after a hyphenated word. To overcome this obstacle, I created a helper function called shiftToHyphen modifies the an input array to start at the hyphenated word which was most recently processed. This modified array is then used with strtok after a hyphenated word is processed to keep its position in the array. Using strtok and this specific method allowed me to process words as soon as I could, to fulfill the bonus portion of the assignment.
2. The majority of my render function is located within a while loop and it uses strtok to process individual words. I have 4 helper functions that I created: "bool hasSpecialCharacter" returns true if a word ends in a double space character and false if it doesn't. "bool hasHyphen" returns true if a word has a hyphen and false if it doesn't. "void separateHypen" separates a hyphen word into it's correct word portions within a C string. "void shiftToHyphen" is used to shift an input array to start at the most recently processed hyphenated word.

Int render

Return 2 if line length less than 1

Initialize all relevant bools (specified later in code)

Initialize all character arrays for storing and copying C strings (specified later in pseudocode)

Initialize character pointers for strtok function

Initialize line counter and hyphen counter to 0

While there are lines in input

Copy line to shiftLine C String and linecopy C string

Get first word

While word isn't null

If word isn't null

If word contains a hyphen and is first word or is following a word

Determine how many hyphens in word. Get first word portion of hyphenated word

while hyphenated word isn’t null

notempty bool set to true

if hyphen word is first word

if less than line length, output and increment line counter or output between lines if greater

if word is not first word

output paragraph break if applicable

if previous word has double space

char and less than line length

Output if no previous break or print word on new line

If space + word can fit on line length

Output if no previous break or print word on new line

If space + word can't fit on line

Output word between lines and split if needed toggling return 1 bool

If word can fit on line

Output word

If word can't fit on line

Output word split between lines and

Set return 1 bool to true

If current word isn't a paragraph break, set has breaked bool to false

If word has a double space character

Toggle double space bool accordingly

Get next word portion of hyphenated word

Sets normal word bool to true to indicate hyphenated word has fully finished processing

Else

Set hyphen counter to number of hyphens in word

If current word is paragraph break

If not first word or has breaked previously, process paragprah break

If first word on line

Output word if shorter than line length or split between lines if longer

If not first word

Process paragraph breaks

If following double space character and less than line length

Process word with double space if hasn't breaked

If following double space character and greater than line length

Process word by splitting through lines if needed

If following word space is less than line length

Process words with space

If following word space is greater than line length

Output word by splitting through lines if needed

If word is a paragraph break

Set has breaked bool to false

If word has double space character

Set endchar bool accordingly

If last word is a hyphen

Use shiftToHyphen function to move C String so strtok can continue working

Get next word with strtok

If notempty bool toggled to true

Output new line

If return 1 bool is toggled return 1 or else return 0

bool hasSpecialCharacter

through to find the latest occurrence of double space character

If index at length-1 is double space character return true else return false

bool hasHyphen

Loop through word and if it has hyphen then return true

Else return false

Void separateHyphen

Loop through word to find instances of – use hyphen counter to keep track of number

Loop through word

If char at index is not a hyphen copy char

Else add a space after hyphen word

Void shiftToHyphen

Loop through word to find first instance of hyphen

Loop through word to find start of word with hyphen

If C string at hyphen index equals a space character while looping

Set index to start at hyphenated word

Loop through C string and transfer characters including the last specified hyphenated word and onwards.

1. Test Data:
2. Tests leading and trailing spaces as well as multiple spaces, new lines, and tabs between words to make sure whitespace is eliminated

testRender(7, " This\n\t\tis a\ntest \t\n ", "This is\na test\n", 0);

1. Tests normal paragraph break

testRender(5, "Hi @P@ hi", "Hi\n\nhi\n", 0);

1. Tests that a trailing paragraph break will not create new lines

testRender(8, "This is a test @P@", "This is\na test\n", 0);

1. Tests that consecutive paragraph breaks only result in on break being activated

testRender(5, "Hi- @P@ @P@ @P@ hi-", "Hi-\n\nhi-\n", 0);

1. Tests that function returns 2 if line length is less than 1

testRender(0, "hi", "hi\n", 2);

1. Tests that function with a line length greater than 250

testRender(300, ". . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .", ". . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .\n", 0);

1. Tests that if a word is longer than line length then function splits across lines and returns 1

testRender(6, "Testing", "Testin\ng\n", 1);

1. Tests that if two words are longer than line length are in a row then they split lines correctly with word portions filling up as much as they can on each line by themselves

testRender(6, "Testing Testing", "Testin\ng\nTestin\ng\n", 1);

1. Tests if two words fit on line length then a space separates both words

testRender(20, "Testing Testing", "Testing Testing\n", 0);

1. Tests a short word that fits on line length starts with a space when following a word that goes over line length

testRender(6, "Testing hi", "Testin\ng hi\n", 1);

1. Tests that when the last character of an output word is a special character, (., ?, !, :) the following word is separated by two blanks

testRender(15, "Testing! hey", "Testing! hey\n", 0);

1. Tests that when the file is empty, then nothing is changed in the file and 0 is returned

testRender(10, "", "", 0);

1. Tests when a hyphenated word fits on line length, then word portions are not split up

testRender(10, "so-called", "so-called\n", 0);

1. Tests when a hyphenated word does not fit on line length then word portions are split up correctly by word portions using hyphen rules

testRender(5, "so-called", "so-\ncalle\nd\n", 1);

1. Tests that a word with multiple hyphens in a row, leading hyphens, and trailing hyphens is split up correctly by word portions using hyphen rules

testRender(3, "-I-love--CS-31-", "-I-\nlov\ne--\nCS-\n31-\n", 1);

1. Tests word with only hyphens

testRender(6, "--------------", "------\n------\n--\n", 0);

1. Tests input with multiple hyphenated words and regular words seperated by spaces to make sure normal world behavior and hyphenated word behavior works together correctly

testRender(8, "Hey- computerscienc--e thirtyone is -SUPER-- Fun--", "Hey-\ncomputer\nscienc--\ne\nthirtyon\ne is -\nSUPER--\nFun--\n", 1);

1. Tests if a leading paragraph break will not create a new line

testRender(10, "@P@ cat", "cat\n", 0);

1. Tests if a word will enter a new line when it is too big for line length following another word

testRender(5, "cat cat", "cat\ncat\n", 0);

1. Tests if a word will enter a new line when it is too big for line length following a special double space character in another word

testRender(5, "cat. cat", "cat.\ncat\n", 0);

1. Tests if a words will split with spaces when hyphenated words are following each other

testRender(5, "a-a -a-a-aa-", "a-a -\na-a-\naa-\n", 0);

1. Tests if the only word in the input ends in a double space character then no spaces are outputted

testRender(10, "computer.", "computer.\n", 0);

1. Tests that a paragraph break wont work when not followed by whitespace before and after

testRender(10, "P@P@P", "P@P@P\n", 0);

1. Tests paragraph break on a new line

testRender(10, "cat.\n@P@\ncat", "cat.\n\ncat\n", 0);

1. Tests consecutive leading paragraph breaks and consecutive trailing paragraph breaks

testRender(10, "@P@ @P@ @P@ cat @P@ @P@ @P@", "cat\n", 0);

1. Tests if only thing in file is paragraph break

testRender(10, "@P@", "", 0);

1. Tests if only thing in file is consecutive paragraph breaks

testRender(10, "@P@ @P@", "", 0);