Problem: Reverse the Words of a Sentence

Level: Easy

Given a sentence, reverse the words of the sentence.

For example,
"i live in a house" becomes "house a in live i"

Questions to Clarify:
Q. What about punctuations (.,)?
A. Assume there are no punctuations.

Q. How to deal with capitalization (uppercase vs lowercase letters)?
A. Ignore the case, keep as it is.

Q. Can I allocate a new String for the result?
A. Yes, you can allocate a new String. Limit the space complexity to O(n).

Note:

There is another solution that doesn't need extra space. It is covered later on in *"Arrays and Strings Part II - Special Tricks"* section.

Keep in mind that Strings are immutable in Java. That means they cannot be modified.
So you will allocate a new String anyway - unless the input is an array of chars.
In other languages, you can actually modify the Strings.

Solution:
We start from the back. As soon as we find a space separating words, we add that word to the result.

Pseudocode:

```
(Note: Never write pseudocode in an actual interview. Unless you're writing a
few lines quickly to plan out your solution. Your actual solution should be in
a real language and use good syntax.)

result = empty string
current_word_end = str.length

i -> str.length-1 to 0
   if str[i] is a space:
     extract str[i..current_word_end], add it to result
     reset current_word_end to i;

// first word was not added by the above loop
```

```
at the end, add the first word - str[0..current_word_end] to result
```

Test Cases:
Corner Cases: null string, empty string, single letter, single space, begins with space, ends with space
Base Cases: one word, two words
Regular Cases: 5 words

Time Complexity: O(n)

Space Complexity: O(n)

Note: We use StringBuilder in Java to append Strings. in Java, just adding to Strings with a '+' creates a new String, which is inefficient (strings are immutable).
This also impresses interviewers. In other languages, this might not be the case. Check if strings are immutable in your language.

```
public static String reverseWords(String s) {
    StringBuilder builder = new StringBuilder();
    int currentWordEnd = s.length();

    for (int i = s.length() - 1; i >= 0; i--) {
        if (s.charAt(i) == ' ') {
            if (builder.length() > 0) // not empty, add a space
                builder.append(' ');

            builder.append(s.substring(i+1, currentWordEnd));
            currentWordEnd = i;
        }
    }

    // add first word
    String firstWord = s.substring(0, currentWordEnd);
    if (builder.length() > 0)
        builder.append(' ');
    builder.append(firstWord);

    return builder.toString();
}

/*
 * Notice that above, you used the following lines twice:
 *
 * if (builder.length() > 0)
 *     builder.append(' ');
 * builder.append(firstWord);
 *
```

```
* It is good to point that out. Mention that in the real world, you could
* extract them into a separate function.
*/
```