=========================
Problem 2a:
=========================
1. set the maximum number of real words found to 0
2. set the best shift to 0
3. for each possible shift from 0 to 27:
4. shift the entire text by this shift.
5. split the text up into a list of the individual words.
6. count the number of valid words in this list.
7. if this number of valid words is more than the maximum number of real words, then
8. record the number of valid words.
9. set the best shift to the current shift
10. increment the current possible shift by 1. Repeat the loop starting at 3.
11. return the best shift.

=========================
Problem 4a:
=========================
There are a couple key points to get in order to understand the solution.

1. The shifts always occur on the first letter of a word in the plain text.
2. The shifts always occur in ascending start positions. So the first shift can't start on the 5th word and the second shift can't start on the 2nd word.
3. The shifts are layered. So the third shift builds on the second shift, which builds on the first shift.
4. Picture how these shifts layer on each other:

```
 plaintext: a very short string
    shift1: 3.................
    shift2:        6..........
    shift3:              4.....
ciphertext: dcyhuacaqx biefdv t
```

which is the string "a very short string" Caesar shifted by 3 starting at "a", Caesar shifted by 6 starting at "short", and Caesar shifted by 4 starting at "string".

Now, consider what would happen if we reverse ONLY shift1. That is, we apply a Caesar shift in reverse - instead of adding 3, we subtract 3.

```
 partial decipher: dcyhuacaqx biefdv t
  reverse shift1:-3.................
 partial decipher: a very ynuxzfbcasxq
```

As you can see, the encryption is undone from the start of the first shift to the start of the second shift. Let's take this new partial decipher and apply a reverse shift2, but only to the part that comes after the first shift.

```
partial decipher: ynuxzfbcasxq
  reverse shift2:-6..........
```

partial decipher: short wxvmrk

Let's repeat for shift3 in the same manner:

partial decipher: bcasxq
  reverse shift3:-4.....
partial decipher: string

If we take all the decoded bits and combine them together, then we get the original plaintext.

This suggests that we can use a variant of our previous algorithm in 2a. We're going to try all possible shifts. But we realize that performing a shift at a particular spot in the string may have one of two outcomes: it may produce spaces between the start of the shift and the end of the string or it may not, in which case, the shifted text should be a word.

5. If a section of the string is shifted by all possible shift values, and no spaces are produced between the start of this section and the end of the string, this section of text is the last word in the sentence. If no valid words are found, then this means there was a bad shift attempted.

The following pseudocode outlines the algorithm:

```
1. for every possible shift from 0 to 27
2. set s to be (the text up until just before the start parameter) concatenated with (the text after the start parameter shifted by the current possible shift)
3. look for spaces beginning at the location specified by the start parameter
4. if there was a space found and the characters from the start location to the location where the space was found form a valid word then
5. we recursively run this same algorithm on the same text, but starting at the location where the space was found.
6. If this recursive call to the function produces a list of tuples, then that means one of the recursive calls found the last word properly and we simply prepend a tuple with the start parameter and the current shift tried to the list and return it.
7. If this recursive call to the function produces None, then that means that there was not combination of shifts and positions found that could produce a valid word at the end, so the error in the shift must either be in the current call to the function, or an earlier call. Continue trying all possible shifts (line 1)
8. if there was NOT a space found and the characters from the start location to the end of the string form a word, then that means we have found a valid shift on this call, so return a list containing a tuple containing the start parameter and the current shift.
9. if there was NOT a space found and the characters from the start location to the end of the string do NOT form a word, then we need to try another shift in this call.

2. If we exhaust all possible shifts and still find no solution from other recusive calls to this function, then that means our start position or a previous shift, is incorrect. Return None.
```