CS118 Program
Hangman, Part B

This program extends Hangman, Part A. You will be starting this program using the code from Part A:

Prepare a function `get_letter()` that has the user guess a letter in the word. The argument to the function from the main program is the `used_letters` list. Have the user provide a guess for a letter, then convert it to an uppercase letter using the string `upper()` method. Use the `in` method on the `used_letters` list to determine if the letter has been guessed already. Collect a letter from the user until the letter provided is not in the `used_letters` list – when that occurs, add the letter to the `used_letters` list. The function doesn't return until a valid letter is gotten from the user, and then there are two return values: the letter provided by the user, and the updated `used_letters` list.

Prepare another function `look_for_match()` that is called upon return from `get_letter()`. The purpose of `look_for_match()` is to look in `word` for the user-provided letter. We'll have this function find each match of the letter and replace each of the corresponding underscores in the `matched_letters` list. We'll also update the `matched_count` variable in this function, so the arguments to this function end up being: the user-provided letter, the `matched_letters` list, `matched_count`, and the `word`, those are the arguments to the function.

The `find()` method in its simple form locates the first occurrence of a substring. But we can use a more complicated form and have it find the first occurrence of a substring *from a given starting position*. To do so, the form is:

```
index = s.find(string_to_find, start_from_index)
```

For example, if we want to find the letter E in the word ANTELOPE:

```
s = 'ANTELOPE'
i = s.find('E', 0)      # finds the first e
j = s.find('E', i+1)    # finds the first e after position i
```

In your function:

- Create a variable `pos` setting its initial value to 0. This will hold the index where the letter is found.
- Define another variable `last_pos` so that it has the value -1. This is the position of the last letter we matched.
- Use the `find()` method so that it searches for the letter starting at `last_pos+1` and saves its result in the variable `pos`.
- Immediately after the method call, give `last_pos` the value that is stored in `pos` – this will allow us to add a loop to search for the next letter.

If this runs once for you, add a loop that will repeatedly find the letter. Since the `find()` method returns -1 when it hasn't located a matching string, have it continue as long as the position returned by the `find()` method is greater than -1. Each time that it finds a letter in word that matches the user-provided letter, do two things:

- Add one to `matched_count` so that we know how many of the words letters have been matched
- Change the corresponding underscore in `matched_letters` to the letter itself.

If the guess does not match a letter, print a consolation string ("I'm sorry – no match"), but don't increment `matched_count`.

The function should return the updated `matched_count` value and `matched_letter` list.

Upon return from the function show the user the `matched_letters` array using your `print_list()` function from Part A.

```
>>>
Testing: The word is SUPERCALIFRAGILISTICEXPIALIDOCIOUS
Your guess? i
After your guess:: _ _ _ _ _ _ _ _ I _ _ _ _ I _ I _ _ I _ _ _ _ I _ _ I _ _ _ I _ _ _
>>>
```