Task 1:

For this task, we'll write a function called `find_emails(text)` to identify and print email addresses found in the `text`. We'll use a simple rule for identifying email addresses – an email is any sequence of characters, separated by spaces, that contains both an at sign (`@`) and a period (`.`). So, things like bob@email.com fit our rule of what an email address looks like. Note that a string might have *multiple* email addresses in it, such as in the following string: *"Contact us at info@example.com or support@company.com for assistance."*

The signature of this function is `(string) ---> None`.

Task 2:

For this task, we'll practice doing a bit of string formatting:

(a) Print your first name, aligned to the left in a width of 10 characters, and your last name, aligned to the right in a width of 20 characters
(b) Print ~~!!~~ centred in a width of 10 characters
(c) Print 17/6, formatted to 3 decimal places
(d) Print 37 in binary (you might have to look this one up)

Task 3:

Another buy-one-get-one free deal! Here, we'll write two functions to practice string manipulation.

The first function, `removePunctuations(text)`, will take a string as input and remove the punctuation marks `?!.,;:"'`.

For example, the input `"Hello, world!"` should return `"Hello world"`, and `"Python's fun, right?"` should return `"Pythons fun right"`.

Make sure to test your function with different sentences containing punctuation to ensure it works as expected.

The signature of this function is `(string) ---> string`.

Next, we'll build a function, `find_and_replace(text, old_word, new_word, case_insensitive=True, word_boundary=True)`. This function will replace occurrences of `old_word` in the text with `new_word`.

However, it's a bit more complicated than that – you can't just return `text.replace(old_word, new_word)`.

First, we want to support case-insensitive replacements. If case_insensitive=True, replacing "fox" with "wolf" in the sentence "The quick brown Fox jumps over the lazy dog." should return "The quick brown wolf jumps over the lazy dog."

Next, if you set `word_boundary=False`, replacing "fox" in "foxes" should yield "wolves". Similarly, replacing "fox" in "fox!" should replace the word without affecting the punctuation.

Note: This task is *tricky!* I would strongly encourage you to build up a solution a bit at a time – first, just handle replacing words with no special handling of these additional requirements. Next, add support for case-insensitive replacement. Then, try to get the word_boundary behaviour to work properly. Be sure to test different combinations of input, considering punctuation and different cases, and don't feel discouraged if you don't get it all working perfectly.

Examples:

```
text = "The quick brown fox jumps over the lazy dog."
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=True))
    # output: The quick brown wolf jumps over the lazy dog.

    text = "The quick brown Fox jumps over the lazy dog."
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=False, word_boundary=True))
    # output: The quick brown Fox jumps over the lazy dog.


    text = "The quick brown fox, jumps over the lazy dog."
    old_word = "fox"
    new_word = "wolf"
```

```
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=True))
    # output: The quick brown wolf, jumps over the lazy dog.


    text = "The foxes are clever."
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=True))
    # output: The foxes are clever.


    text = "The foxes are clever."
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=False))
    # output: The wolfes are clever.


    text = "The fox! ran away."
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=True))
    # output: The wolf! ran away.


    text = "The fox and the Fox were friends."
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=True))
    # output: The wolf and the wolf were friends.


    text = "The fox ran after the rabbit. Another fox watched."
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=True))
    # output: The wolf ran after the rabbit. Another wolf watched.
```

```
    text = ""
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=True))
    # output: empty string


    text = "The cat runs away."
    old_word = "fox"
    new_word = "wolf"
    print(find_and_replace_advanced(text, old_word, new_word,
case_insensitive=True, word_boundary=True))
    # output: The cat runs away.
```