Swift Exercise
Functions

Exercise — Look-and-Say Numbers

In this exercise, we want to generate sequences of “look-and-say numbers”. These are sequences where, each time, we apply the “look-and-say” (=“read aloud”) operation to obtain the next number. The “look-and-say” operation consists in “reading out” (from left to right) the sequences of digits of the number. It’s better explained with some examples:

- 1 is read as "One 1", therefore it becomes 11;
- 11 is read as "Two 1s", therefore it becomes 21;
- 21 is read as "One 2 and one 1", therefore it becomes 1211;
- 1211 is read as "One 1, one 2 and two 1s", therefore it becomes 111221;
- 111221is read as "Three 1s, two 2s and one1", therefore it becomes 312211;
- and so on...

Therefore, the resulting number of applying 5 times the look-and-say property to 1 is 312211. In short, this property gives us something like

$$N_{\text{consecutive same digit } d_1} d_1 \cdots N_{\text{consecutive same digit } d_k} d_k$$

When operating on a number d1 · · · dk , where k is the number of digit *changes*. For instance, with the number 111221 (where k = 3, d1 = 1, d2 = 2 and d3 = 1), we have:

312211

We would like to write a program that can apply, a given number of time, the “look-and-say” operation to some given number. In this assignment, we will limit our tests to a very few number of repetitions, on small numbers, so as not to overflow the (Int) type representation capacity.

Methodology

First, we need to be able to operate on the digits of a number from left to right in order to be able to “read it out loud”. For this, we need a function to get us the leftmost digit of a number and remove it from that number so that we can go on, operating on the next digit.
The provided function separate_digit_left takes a number and modifies it, removing its leftmost digit which is returned (as return value). If some variable x is 1234, separate_digit_left (x) returns 1, and x has been modified to 234.
When we have this, we need some other functions to apply get the “look-and-say” sequence from a given number:

- (function add_digit_right, which means “add/push right digit”) so as to create the new number in the sequence, we need to be able to add a digit to the right of some number; for instance, if x is 1234, then add_digit_right (x, 5) modifies x into 12345;
- (function say_digit, which means “say digit”) we also need to be able to add to the right of some number, a digit as well as its number of occurrences; for instance add « One ’1” », i.e. « 11 », to the right of 3122 (thus getting 312211); this can easily be achieved with two calls to the former function add_digit_right;
- (function look_and_say, which means “look/read and say”) we furthermore need to be able to apply *once* the “look-and-say” operation to some given number; this can be done by:
    - Separate a first time the left digit(function separate_digit_left);
    - repeat until the manipulated number is null:
        - separate once more the left digit;
        - if it’s the same as before, increase its number of repetitions;
        - otherwise add to the result, the preceding digit with its number of repetitions (function say_digit);
    - properly initialize and properly update all the intermediate variables needed;

- (function repeat_look_and_say, provided ,it means “repeat look/read and say”) finally, we need to apply *several times* the “look-and-say” operation to some given number.

Execution Example

Your program will take 2 numbers from standard input: the first number and the number of times to apply the property, such as

```
1    5
```

and output the resulting number. For the above input, the correct program would output the following:

```
312211
```

Notice: There will be no 0 digits in the input.

Attention: When testing, try not to provide too large numbers as input, such as more than one digit to the first number and more than 6 as the number of times.