Algorithmics
Final Exam #1 (P1)

Undergraduate 1st year S1
Epita

7 January 2020 - 13h-15h

Instructions (read it) :

- ☐ You must answer on **the answer sheets provided.**
  - No other sheet will be picked up. Keep your rough drafts.
  - Answer within the provided space. **Answers outside will not be marked**: Use your drafts!
  - Do not separate the sheets unless they can be re-stapled before handing in.
  - Penciled answers will not be marked.
- ☐ The presentation is negatively marked, which means that you are marked out of 20 points and the presentation points (maximum of 2) are taken off this grade.
- ☐ **Code:**
  - All code must be written in the language `Python` (no C, Caml, Algo or anything else).
  - **Any `Python` code not indented will not be marked.**
  - All that you need (types, routines) is indicated in the **appendix** (last page)!
- ☐ Duration : 2h

EVERYTHING FAMILIAR HAS DISAPPEARED! THE WORLD LOOKS BRAND-NEW!
A NEW YEAR... A FRESH, CLEAN START!

Exercise 1 (Abstract Types: Iterative lists (delete) – *3 points*)

Consider the signature of the algebraic abstract type *iterative list* seen in class and included below.

TYPES
list, box

USES
element, integer

OPERATIONS

emptylist	:	$\rightarrow$ list
access	:	list $\times$ integer $\rightarrow$ box
contents	:	box $\rightarrow$ element
nth	:	list $\times$ integer $\rightarrow$ element
length	:	list $\rightarrow$ integer
insert	:	list $\times$ integer $\times$ element $\rightarrow$ list
next	:	box $\rightarrow$ box

The abstract type given above is incomplete. The delete operation has been forgotten...
Its profile is as follows:

OPERATIONS

delete	:	list $\times$ integer $\rightarrow$ list

where *delete*(λ, k) deletes the k^{th} element of the list λ (k begins at 1).

1. Specify the PRECONDITIONS of the operation *delete* if there are any.
2. Give the axioms defining the operation *delete* in all its use-cases.

Exercise 2 (Binary Search "path" – *2 points*)

Assume we are given some lists sorted in increasing order. We want to search for the value 66 using the binary search algorithm in each one of these lists.

Which sequences among the following could be the order of values encountered during the search?

① 46 - 65 - 81 - 73 - 70 - 66

② 31 - 62 - 90 - 72 - 61 - 66

③ 36 - 70 - 53 - 40 - 42 - 66

④ 35 - 51 - 55 - 58 - 61 - 66

Kaprekar Routine

The following exercises are independent. The last one uses functions from the previous exercises, even if they have not been written.
In all functions, we assume that the parameters are valid: there is neither a test to perform nor an exception to raise.

Exercise 3 (Test - *1 point*)

Let the fonction `test` be defined as follows:

```python
def test(x, L):
    i = len(L) - 1
    while i >= 0 and L[i] != x:
        i = i - 1
    return i >= 0
```

What does this function do?

Exercise 4 (Integers ↔ list – *6 points*)

1. Write the function `int_to_list(`n`, `p`)` that converts n (natural number with at most p digits) into a list of its digits, possibly completed by `0` to reach p digits.

   Examples of results (order of digits in the result does not matter):

```
>>> int_to_list(27972, 5)
[2, 7, 9, 7, 2]

>>> int_to_list(42, 5)
[2, 4, 0, 0, 0]

>>> int_to_list(258, 4)
[8, 5, 2, 0]
```

2. Write the function `list_to_ints(`L`)` that returns the pair ($left$, $right$), from the not empty list L that contains only digits (from 0 to 9), such that

   - $left$: is the integer built with the digits "read" from left to right,
   - $right$: is the integer built with the digits "read" from right to left.

   Examples of results:

```
>>> list_to_ints([1, 2, 3, 4, 5, 6])
(123456, 654321)

>>> list_to_ints([2, 4, 0, 0, 0])
(24000, 42)
```

Exercise 5 (Histogram and sort – *4 points*)

Here we work with lists that contain only digits (from 0 to 9).

1. Write the function `hist(`L`)` that returns a histogram (a list) of digits in the list L.

   Reminder: the histogram H is a list such that $H[i]$ is the occurrence number of the value i. For instance in the first application below, there are 5 values 0, 3 values 1, 3 values 2...

   Examples of results:

```
>>> hist([1,5,9,3,0,1,2,0,1,0,2,5,0,5,2,0])
[5, 3, 3, 1, 0, 3, 0, 0, 0, 1]
>>> hist([1,0,1,0,1,0,1,0,1])
[4, 5, 0, 0, 0, 0, 0, 0, 0, 0]
```

2. Use the function `hist` to write the function `sort(`L`)` that sorts the list L in increasing order (the function builds and returns a new list.)

   Example of result:

```
>>> sort([1,5,9,3,0,1,2,0,1,0,2,5,0,5,2,0])
[0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 5, 5, 5, 9]
```

Exercise 6 (Kaprekar – *5 points*)

Kaprekar Routine:

Let n be a p-digit positive integer. For instance, $n = 6264$.

- Arrange the p digits of n in descending and ascending order: 6642 and 2466
- Subtract and possibly add '0' to obtain a p-digit number: n = 6642 - 2466 = 4176.
- Do this again with the result. 4176 → 7641 - 1467 = 6174

The Kaprekar routine can be done with any number. In any case, the sequence **will reach a value already met** (in the examples given later: 6174 is reached twice in the first example, as is 63 in the second one), that will be the value 0 if all digits were identical.

Comment:

Here, we always compute p-digit numbers. That is, if a result is less than 10^{p-1} (for example 999 when $p = 4$), the two new numbers will be build from the number digits completed by 0 (9990).

The function you have to write `Kaprekar(n, p)` takes a natural number n as parameter as well as its number of digits p. It performs the Kaprekar routine. It builds and returns the list of the computed values (until we find a value already met) as in the following examples.

Examples of results:

```
>>> Kaprekar(1574, 4)
[1574, 6084, 8172, 7443, 3996, 6264, 4176, 6174, 6174]   # 6174 met twice

>>> Kaprekar(42, 2)
[42, 18, 63, 27, 45, 9, 81, 63]   # 63 met twice

>>> Kaprekar(666, 3)
[666, 0, 0]    # 0 met twice
```

You can use the functions of the exercises 3 to 5, even if they have not been written down.

Appendix: Authorised functions and methods

You can use the methods `append` and the function `len` on lists, as well as the function `range` :

```
    >>> L = []

    >>> for i in range(5):
             L.append(i)

    >>> L
    [0, 1, 2, 3, 4]

    >>> len(L)
    5
```

No operators are allowed on lists (`+, *, ==` ...).

Your functions

You can write your own functions as long as they are documented (we have to know what they do).

In any case, the last written function should be the one which answers the question.