Algorithmics
Final Exam #1 (P1)

Undergraduate 1st year (s1)
Epita

3 Jan. 2017 - 10 : 00

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

Garage and Research

Exercise 1 (Stacks and garage – *3 points*)

Imagine a garage with two entrances and one exit (see figure 1). To park your car there are two access paths (*entrance e1* and *entrance e2*) and to exit the garage there is a single path (*exit s*).

The possible movements of the cars are:

- a car can enter by the *entrance e1*
- a car can enter by the *entrance e2*
- a car can exit only by the *exit s*
- A car can exit only if it is the last entered (regardless of entrance)
- A car cannot jump over another car

Remark: There is no problem of cars coming face to face.

The garage works like a stack with two inputs.

Symbolize an entry using the couple $(v_{car\ number}, e_{entrance\ used\ number})$ and the exit by the letter s.

The following sequence of actions is carried out:
(v_1, e_1),(v_2, e_1),(v_3, e_2),(v_4, e_1),s,s,(v_5, e_2),(v_6, e_2),s,s,s,(v_7, e_1),(v_8, e_2),(v_9, e_2),s,s,s,s

Therefore, cars exit the garage in the following order:
v_4,v_3,v_6,v_5,v_2,v_9,v_8,v_7,v_1

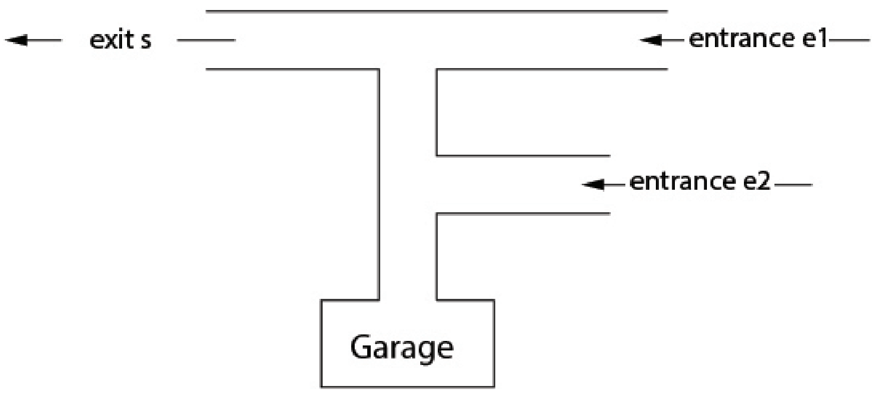

Figure 1: garage with entrances and one exit.

1. Are the following vehicle input/output sequences valid?

   (a) (v_1, e_1),(v_2, e_1),(v_3, e_1),s,s,(v_4, e_2),(v_5, e_1),s,s,s,(v_6, e_2),s

   (b) (v_1, e_1),(v_2, e_2),s,(v_3, e_2),s,s,s,(v_4, e_1),(v_5, e_2),s,(v_6, e_1),(v_7, e_2),s,s

   If the sequence is invalid, briefly state why.

2. We can symbolize the action "enter a car" by the symbols $E1$ or $E2$ (*push*) depending on the used entrance and the action "exit a car" by the symbol D (*pop*).

   The sequence of actions presented at the beginning can then be symbolized by the following sequence:

   $$E1E1E2E1DDE2E2DDDE1E2E2DDDD$$

   Give a rule that would characterize the admissible sequences.

Exercise 2 (Binary Search: 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 not be** the order of values encountered during the search? On the answer sheets, circle the impossible sequences.

① 46 - 65 - 81 - 73 - 70 - 66

② 31 - 62 - 90 - 72 - 61 - 66

③ 36 - 70 - 53 - 50 - 61 - 66

④ 35 - 51 - 55 - 58 - 61 - 66

Python: Kaprekar Process

The following exercises are independent EXCEPT the last one that uses functions from the previous exercises.
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 – *5 points*)

1. Write the function `int_to_list`(n, p) that converts n (naturel 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]
```

2. Write the function `list_to_ints`(L) that returns the pair ($left$, $right$), from the 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 on 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).

This process can be described by the following "pseudo-algorithm":

```
procedure Kaprekar(integer n, p)

variables
   integer   low, high
   list L
begin
   L ← empty-list
   while n ∉ L do
      add x to L
      low ← number with digits of n in ascending order
      high ← number with digits of n in descending order
      n ← high - low
   end while
end
```

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. Moreover, the different values have to be displayed as in the following examples.

Examples of results:

```
>>> Kaprekar(1574, 4)
1574 -> 6084 -> 8172 -> 7443 -> 3996 -> 6264 -> 4176 -> 6174 -> 6174

>>> Kaprekar(42, 2)
42 -> 18 -> 63 -> 27 -> 45 -> 9 -> 81 -> 63
```

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 method `append` and the function `len` on lists:

```
>>> help(list.append)
Help on method_descriptor:    append(...)
    L.append(object) -> None -- append object to end of L

>>> help(len)
Help on built-in function len in module builtins:    len(...)
    len(object)
    Return the number of items of a sequence or collection.
```

You can also use the functions `range` and `print`. Reminder:

```
>>> for i in range(10):
...     print(i, end=' ')
0 1 2 3 4 5 6 7 8 9

>>> for i in range(5, 10):
...     print(i, end='-')
5-6-7-8-9-
```