Algorithmics
Final Exam #1 (P1)

Undergraduate 1^{st} year S1#
Epita

19 juin 2018 - 9 : 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

BIEN ENTENDU, QUELQUES MINIMES PRÉCAUTIONS S'IMPOSENT... VOUS ÉTONNEREZ-VOUS SI JE VOUS PROPOSE QUELQUES MENUS TESTS ?
BIEN SÛR QUE NON.
TRAPPE

Exercise 1 (Searching algorithms – *3 points*)

Let λ be the following list:

$$\lambda = \{1, 3, 8, 15, 23, 29, 32, 35, 38, 43, 47, 51, 55\}$$

The value 36 is been searched in this list. For each research method, give the number of comparisons between the searched value and a list element.

1. Linear search regardless of element order
2. Linear search taking into account the element order
3. Binary search

Exercise 2 (Séquences et dichotomie – *3 points*)

1. Which sequences among the following could be the order of values encountered during the binary search algorithm in a list sorted in increasing order?
   (a) 50, 70, 2048, 75, 1500, 1024
   (b) 50, 75, 2048, 70, 1500, 1024
   (c) 2048, 50, 70, 75, 1500, 1024
   (d) 50, 75, 70, 2048, 1500, 1024
2. Assume we are given a search sequences as a list. Give the rinciple of an algorithm that tests whether this sequence is valid.

Exercise 3 (- *4 points*)

Consider the signature of the algebraic abstract type *iterative list* (extract) included below.

TYPES
list, box

USES
element, integer

OPERATIONS

emptylist	:	→ list
nth	:	list × integer → element
length	:	list → integer

We propose to extend the properties of this type using the operation *mystery*:

OPERATIONS
mystery : list → list

AXIOMS
$\lambda \neq$ emptylist & $1 \leqslant$k$\leqslant$length(λ) $\Rightarrow$ nth(mystery(λ),k) = nth(λ, length(λ)-k+1)
length(mystery(λ)) = length(λ)

WITH

list	λ
integer	k

1. What is the name of the operation *mystery*?
2. Implement in Pyhon the operation *mystery*. Warning, this function does not return aniting: the list is modified **in place**.

Exercise 4 (What is it? – *3 points*)

Let `what` be defined below:

```
def what(p, v):
    n = len(p)
    if n < 2:
        raise Exception("not enough")
    (a, b) = p[0]
    (c, d) = p[1]
    i = 1
    while i < n - 1 and b < v:
        i += 1
        (a, b) = (c, d)
        (c, d) = p[i]
    return b + (d - b) * (v - a) / (c - a)
```

1. Give the results of the following aplications of `what`:

   (a) `what([(0,0), (10,10), (20, 20), (30, 30)], 15)`

   (b) `what([(0,0), (10,20), (20,40), (30,60)], 24)`

   (c) `what([(0,0), (1, 10), (2,100), (3, 1000)], 2.5)`

   (d) `what([(0,3), (1,6), (2,9), (3,10), (4,15)], 20)`

2. Let L be a list of integer pairs $[(x_0,y_0), (x_1,y_1), \cdots, (x_{n-1},y_{n-1})]$ and Y a number. What does `what(`L, Y`)` compute?

Exercise 5 (Select Sort (Tri par sélection) – *8 points*)

1. Write the function `minimum(`L, d, f`)` that returns the position of the minimum value in the list L between the positions d and f, both included (with $0 \leq d < f < len(L)$).

   For instance, in the following list:

	0	1	2	3	4	5	6	7	8	9
L	4	-5	3	-3	8	-2	0	3	-6	7

   Between the positions $d = 2$ and $f = 7$, the minimum is at position 3.

2. Use the previous function to write a function that sorts a list in increasing order **in place** (the list is modified, no other list should be used).

   Application example:

```
>>> L = [4, -5, 3, -3, 8, 2, 0, 3, -6, 7]
>>> selectSort(L)
>>> L
[-6, -5, -3, 0, 2, 3, 3, 4, 7, 8]
```

3. Let n be the number of elements of the list, give for the selection sort:

   (a) the number of performed comparisons;

   (b) the number of element copies.

Appendix

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 function `range` and `raise` to raise exceptions. Reminder:

Quelques rappels :

```
>>> 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

>>> (a, b) = (1, 2)
>>> (a, b) = (b, a)
>>> (a, b)
(2, 1)
```