Algorithmics
Final Exam #1 (P1)

Undergraduate 1^{st} year S1
EPITA

8 Jan. 2019 - 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

Exercise 1 (Binary Search: search "path" – *2 points*)

Assume we are given some lists sorted in increasing order. We want to search for the value 42 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 50 - 15 - 48 - 22 - 46 - 42

② 48 - 15 - 45 - 22 - 47 - 42

③ 15 - 22 - 45 - 43 - 35 - 42

④ 22 - 45 - 43 - 15 - 35 - 42

Exercise 2 (Searching algorithms – *3 points*)

Let λ be the following list: $\lambda = \{1, 5, 7, 16, 21, 30, 33, 34, 42, 49, 72, 81, 99\}$
The value 40 is been searched in this list. For each research method, give the number of comparisons between the searched value and an element of the list.

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

Exercise 3 (See Syracuse – *3 points*)

The Syracuse sequence is defined below ($n \in \mathbb{N}^*$):

if n is even, the following number will be $n/2$
if n is odd, the following number will be $3n + 1$

Collatz conjecture : $\forall n \in \mathbb{N}^*$ this sequence will eventually reach the number 1.

Write the function `Syracuse` that builds the list of all the values of Syracuse sequence from a nonzero natural number n to 1. If n is not a nonzero natural, the function returns the empty list.

Application example:

```
>>> Syracuse(14)
[14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
```

Exercise 4 (Arithmetic progression – *4 points*)

Write a function that tests whether the integers of a list, containing at least two elements, follow an arithmetic progression.

Reminder: An arithmetic progression is a sequence of numbers such that the difference between two consecutive terms is constant (the common difference).

If the list has at least two elements and follows an arithmetic progression with a nonzero common difference, the fonction returns the common difference of the sequence. In others cases, the function returns 0.

Application examples:

```
>>> arithmetic([1, 2, 3, 4, 5, 6, 7])
1
>>> arithmetic([1, 1, 1, 1, 1])
0
>>> arithmetic([12])
0
>>> arithmetic([5, 8, 11, 14, 17])
3
```

Exercise 5 (Deletion in sorted list – *5 points*)

Write the function `delete(`L`, `x`)` that removes the value x, if it exists, from the list L sorted in strictly increasing order. The function returns a boolean that indicates whether the deletion occurred.

Application examples:

```
>>> L = [1, 4, 5, 7, 8, 12, 15]
>>> delete(L, 7)
True
>>> L
[1, 4, 5, 8, 12, 15]
>>> delete(L, 11)
False
>>> L
[1, 4, 5, 8, 12, 15]
```

Exercise 6 (What is it? – *3 points*)

Let `what` be the following function:

```
def what(L) :
    me = 0
    for e in L :
        if e > me :
            me = e

    X = []
    for i in range(me + 1) :
        X.append(0)
    for e in L :
        X[e] +=1

    LT = []
    for i in range(me + 1) :
        for j in range(X[i]):
            LT.append(i)
    return LT
```

1. Give the result of the following application of `what`:

```
>>> what([1,3,2,8,7,2,5,4,0,6,2,15])
#FIXME
```

2. We call `what(`L`)` with L a list of natural numbers.
   (a) At the end of the first loop, what does `me` represent?
   (b) At the end of the third loop, what does `X` represent?
   (c) What does this function returns?
3. **Bonus:** What is the complexity of this function?

Appendix: Authorised functions and methods

You can use the methods `append`, `pop` (without argument only) 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.

>>> help(list.pop)  # cannot be use with index here...
Help on method_descriptor:
pop(...)
    L.pop() -> item -- remove and return last item.
    Raises IndexError if list is empty.
```

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

You can also use the function `range` and `raise` to raise exceptions. 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

>>> raise Exception("blabla")
...
Exception: blabla
```

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.