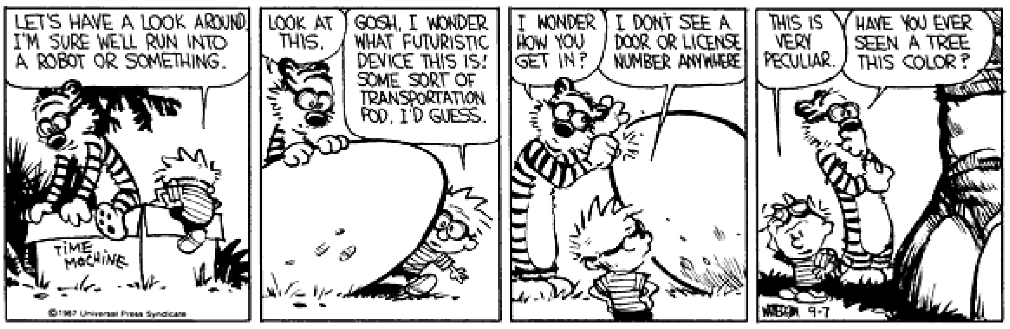
Algorithmics
Midterm #2 (C2)

Undergraduate 1st year S2
Epita

2 March 2020 - 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 (functions, methods) is indicated in the **appendix** (last page)!
- ☐ Duration : 2h


LET'S HAVE A LOOK AROUND. I'M SURE WE'LL RUN INTO A ROBOT OR SOMETHING.
TIME MACHINE
LOOK AT THIS.
GOSH, I WONDER WHAT FUTURISTIC DEVICE THIS IS! SOME SORT OF TRANSPORTATION POD, I'D GUESS.
I WONDER HOW YOU GET IN?
I DON'T SEE A DOOR OR LICENSE NUMBER ANYWHERE
THIS IS VERY PECULIAR.
HAVE YOU EVER SEEN A TREE THIS COLOR?

Exercise 1 (A little coursework. . . – *4 points*)

Assume the general tree **T** is represented in the binary form **left most child - right sibling**:

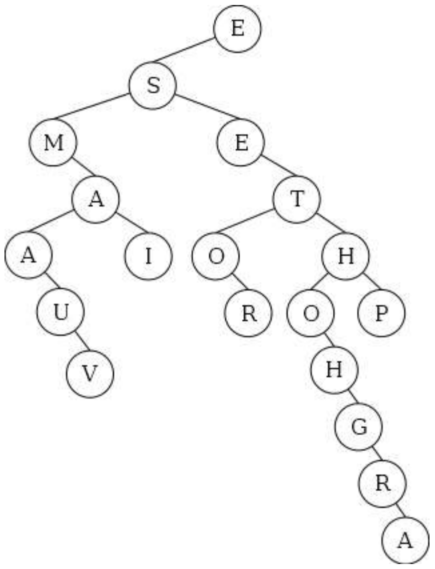
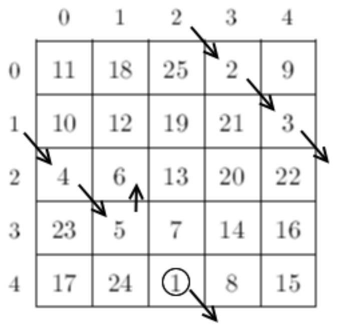


Figure 1: The general tree **T** in binary form **left most child - right sibling**

1. What is the size of the tree `T`?
2. What is the height of the tree `T`?
3. What is the internal path length of the tree `T`?
4. What is the external average depth of the tree `T`?
5. Give the list of nodes of the tree `T` encountered in postorder traversal.
6. Give the list of nodes of the tree `T` encountered in level order.

Exercise 2 (Magic Square – *4 points*)

Definition:
A magic square of order n is an arrangement of n^2 numbers, usually integers, in a square grid, where the numbers in each row, and in each column, and the numbers in the main and secondary diagonals, all add up to the same number.



	0	1	2	3	4
0	11	18	25	2	9
1	10	12	19	21	3
2	4	6	13	20	22
3	23	5	7	14	16
4	17	24	1	8	15

Figure 2: Magic Square of order 5

Construction :
The *Siamese method* allows to build a magic square of odd *order* n that contains all values from 1 to n^2.

- The value 1 is placed on the last line, in the middle;
- Moves for the next values:
  - if the current value is a multiple of n, the next value is in the cell just above,
  - otherwise, the next value is directly below to the right.

Write the function `Siamese(`n`)` that builds a magic square of order n (odd integer greater than 2).

Exercise 3 (Sub-List – *5 points*)

Write the function `sub_line(`M`, `L`)` that checks if the list L is included in one of the lines of the matrix M (assumed non empty).

1	1	10	10	7
10	0	9	3	8
3	1	4	7	5
0	8	1	1	1
4	5	1	8	9
12	11	7	8	1
6	2	10	9	8

Figure 3: `Mat1`

Application examples on the matrix in figure 3:

```
>>> sub_line(Mat1, [1, 10])
True
>>> sub_line(Mat1, [1, 10, 7])
False
>>> sub_line(Mat1,[4, 5, 1, 8, 9])
True
>>> sub_line(Mat1, [7, 8, 1])
True
```

Exercise 4 (Partially ordered tree – *3 points*)

Definition:
A *partially ordered tree* is a labeled binary tree, where in each node the value stored is less than or equal to the values stored in its subtrees. An empty tree is considered as partially ordered.

Write the function `priority(`B`)` that checks if the binary tree B (whose keys are non zero naturals) is partially ordered.

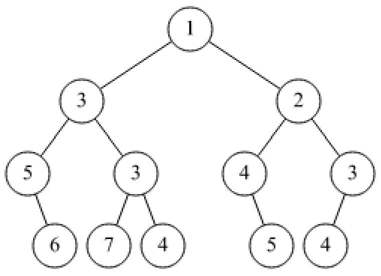


Figure 4: `priority(B1)` → `True`

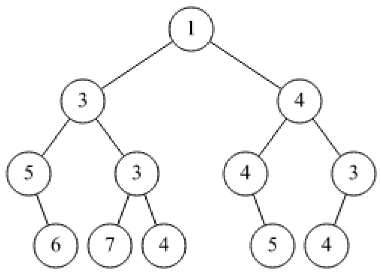


Figure 5: `priority(B2)` → `False`

Exercise 5 (Width – *4 points*)

Definition:
In a tree, the *width of a level* is the number of nodes of this level.
The *width of a tree* is the maximum of the widths of its levels.

Write a function that calculates the width of a binary tree.

Application example on the tree in figure 4:

```
>>> width(B1)
5
```

Appendix

Binary Trees

The binary trees we work on are the same as the ones in tutorials.

- `None` is the empty tree.
- The non-empty tree is an object of class `BinTree` which has 3 attributes: `key`, `left`, `right`.

```
class BinTree:
    def __init__(self, key, left, right):
        self.key = key
        self.left = left
        self.right = right
```

Appendix: Authorised functions and methods

You can also use the function `range`. 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
```

List

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.
```

Matrices

You can use the function `initMat(`$line, col, val$`)` that returns a new matrix of *line* lines and *col* columns filed with *val*.

Queues

The methods of the class `Queue`, assumed imported:

- `Queue()` returns a new queue ;
- q`.enqueue(`e`)` enqueues e in q ;
- q`.dequeue()` deletes and returns the first element of q ;
- q`.isempty()` tests whether q is empty.

Your functions

You can write your own functions as long as they are documented (we have to known what they do).
In any case, the last written function should be the one which answers the question.