Algorithmics
Final Exam #3 (P3)

Undergraduate 2nd year (S3)
Epita

16 December 2016 - 9 : 30

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 (classes, types, routines) is indicated where needed!
- 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.

☐ Duration : 2h

SANTA WON'T COME UNTIL WE'RE ASLEEP, YOU KNOW.
I CAN'T TAKE THE SUSPENSE!

Some Graphs

Exercise 1 (Connections by and large... – *4 points*)

1. In a meeting of n people, is it true that there are always at least two people with the same number of relationships (it is assumed that if A knows B, then B knows A)?

   Justify your answer.

2. Let X be a set of rabbits and G a directed graph with X for the vertex set. We say that G is a ***kinship graph*** if the arcs $x \rightarrow y$ of G code the relation y *is the child of* x and if there is only one *Adam* and only one *Eve*. Give three conditions that G must necessarily check to be a kinship graph ?

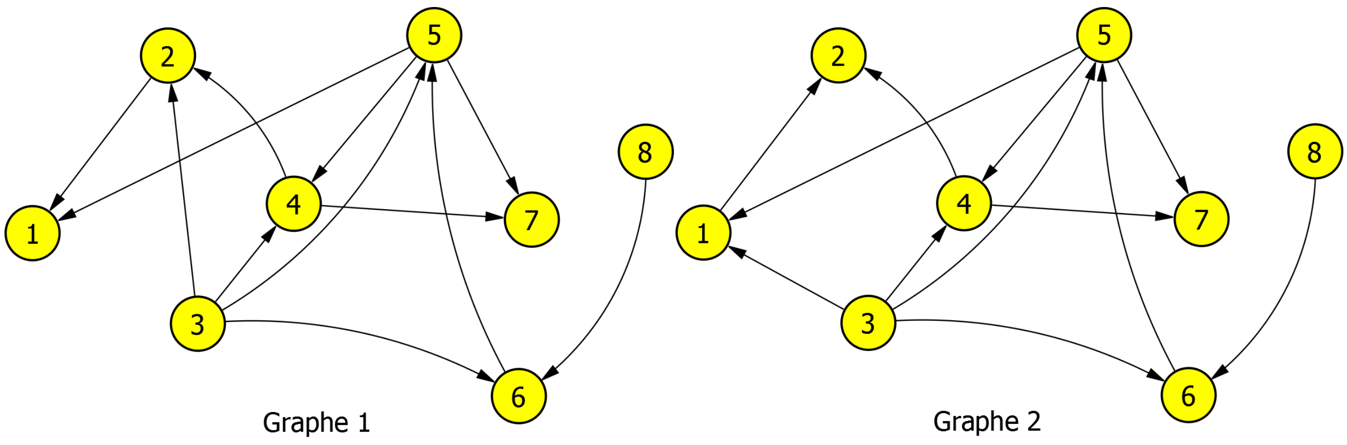

Figure 1: Two *kinship graphs*?

3. Which graph(s) in picture 1 deserve(s) the designation of ***kinship graph***?

Exercise 2 (Implementation and questions... – *2 points*)

Imagine an undirected graph G of 7 vertices represented by the following adjacency matrix:

	1	2	3	4	5	6	7
1		T	T				
2	T			T	T		
3	T					T	T
4		T					
5		T					
6			T				
7			T				

1. Give two graph properties that the transitive closure of G has.

2. Give the Depth-First Search postorder list of vertices of G. As usual, you will start from the vertex 1 and assume that the successors are met in increasing order.

Some Trees

Exercise 3 (Red-Black Trees and Mystery – *3 points*)

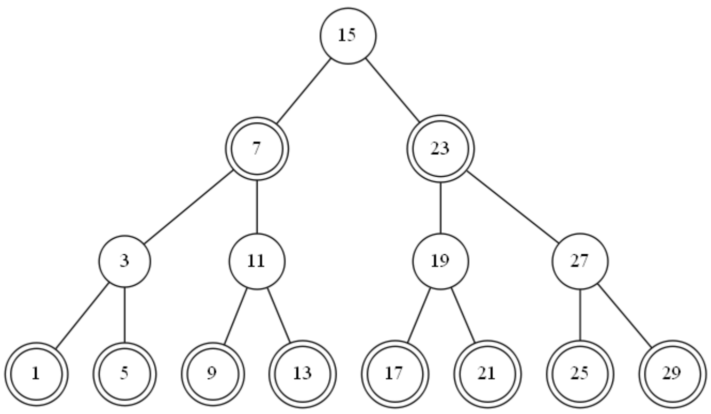

Figure 2: Tree B_1

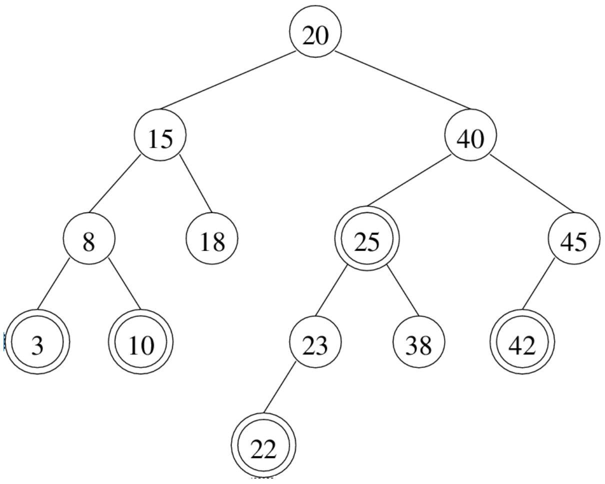

Figure 3: Tree B_2

Remark: The red nodes are those with "double circles".

```python
def __what(T):
    if T == None:
        return (0, 0)
    else:
        (a, b) = (1, 1)
        if T.left and T.left.red:
            (a1, b1) = __what(T.left.left)
            (a2, b2) = __what(T.left.right)
            a += a1 + a2 + 1
            b += b1 + b2
        else:
            (a1, b1) = __what(T.left)
            a += a1
            b += b1
        if T.right and T.right.red:
            (a1, b1) = __what(T.right.left)
            (a2, b2) = __what(T.right.right)
            a += a1 + a2 + 1
            b += b1 + b2
        else:
            (a1, b1) = __what(T.right)
            a += a1
            b += b1
        return (a, b)

def whatIsThis(T):
    (a, b) = __what(T)
    return (a / b)
```

1. Let B_1 and B_2 be the trees in figures 2 and 3. For each of the following calls:
   - what is the result?
   - how many calls of `__what` have been done?

   (a) `whatIsThis(`B_1`)`
   (b) `whatIsThis(`B_2`)`

2. Let A be the 2-4 tree represented by the red-black tree B. What is the measure on A given by `whatIsThis(`B`)`?

Some Graphs that are Trees

In the next exercises, the graphs we work on are undirected and represented by adjacency lists.

Exercise 4 (I want to be tree – *5 points*)

Définition :
A **tree** is an **acyclic connected** graph.

Write the function `isTree` that tests whether a graph is a tree.

Exercise 5 (Diameter – *6 points*)

Définitions :

- The **distance** between two vertices in a graph is the number of edges in a **shortest path** connecting them.
- The **diameter** of a graph is the **highest distance** between any pair of vertices.

When the graph is a tree, computing the diameter is simple:

- From any vertex s_0, find a vertex s_1 whose distance from s_0 is maximal.
- From s_1, find a vertex s_2 whose distance from s_0 is maximal.
- The distance between s_1 and s_2 is the graph diameter.

Write the function `diameter` that computes the diameter of a graph that is a tree.

Appendix

Classes, functions and methods can be used without prefix.

Graphs

```python
class Graph:
    def __init__(self, order, directed = False):
        self.order = order
        self.directed = directed
        self.adjLists = []
        for i in range(order):
            self.adjLists.append([])
```

Queues

- `Queue()` returns a new queue
- `enqueue(`e`, `q`)`
- `dequeue(`q`)`
- `isEmpty`q`)`

Others

- on lists: `len`
- `range`.
- `min` and `max`, but only with two integer values!

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.