Algorithmics
Midterm #3 (C3)

Undergraduate 2^{nd} year - S3
EPITA

29 October 2018 - 13 : 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 (class, types, routines) is indicated in the appendix (last page).
  - 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

JE CROIS QUE QUELQU'UN M'A DEMANDÉ QUAND COMMENCERONT NOS AVENTURES...
ELLES COMMENCERONT DÈS QUE VOUS AUREZ PRIS LA PEINE DE TOURNER CETTE PAGE!... JE VOUS REMERCIE POUR VOTRE ATTENTION!

Exercise 1 (Hashing Strongly Connected – *4 points*)

1. Give a direct method of hashing.
2. Which collision resolution method does not need a hash table whose size is greater than the number of keys to be hashed?
3. Which kind of search is incompatible with the hashing?
4. With which collision resolution method do secondary collisions appear?

Let `G=<S,A>` be a directed graph defined by:

```
   S={1,2,3,4,5,6,7,8,9,10}
and A={(1,2),(1,6),(1,7),(2,3),(2,6),(3,1),(3,5),(4,3),(4,8),(4,9),(4,10),
      (5,1),(7,6),(8,5),(8,10),(10,9)}
```

5. Draw the corresponding graph to `G`.
6. Give the table `InDegree` such as $\forall\ i \in [1, Card(S)]$, `InDegree[i]` is equal to the indegree of `i` in `G`.

Exercise 2 (Equality – *5 points*)

Write the function `same(`T`,` B`)` that tests whether T, a general tree in "classical" representation, and B, a general tree in *first child - right sibling* representation, are identical. That is, they contain same values in same nodes.

Exercise 3 (Levels – *4 points*)

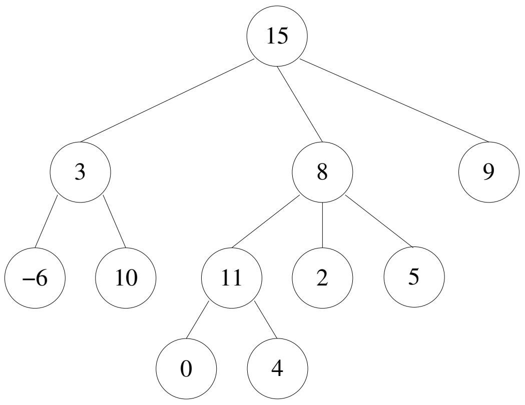

Figure 1: T

Write a function that builds a list of the keys of a tree in *first child - right sibling* representation level by level: a list of lists, each sub-list contains the keys of a level.

Application example, with `T` *the tree in figure 1:*

```
>>> levels(T)
[[15], [3, 8, 9], [-6, 10, 11, 2, 5], [0, 4]]
```

Exercise 4 (Maximum Gap – *4 points*)

For each node of a B-tree, we call *gap* the maximum difference between two successive keys. Le *maximum gap of a B-tree* is the maximum *gap* of its nodes.

For instance, in the tree in figure 2, the maximum gap is 16 (42 - 26).

Write the function `maxgap` that computes the maximum gap of a B-tree. The function returns 0 for an empty tree.

Exercise 5 (B-Trees and Mystery – *3 points*)

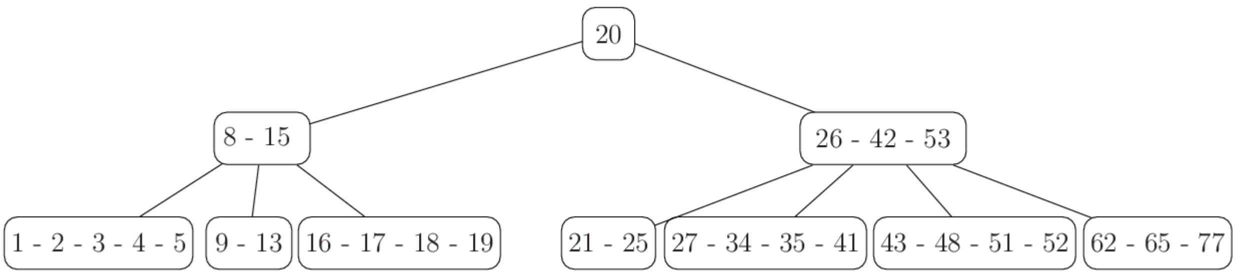

Figure 2: B-tree B_1

```python
def mystery(B, a, b):
    i = 0
    while i < B.nbkeys and B.keys[i] < a:
        i += 1
    c = 0
    if B.children == []:
        while i < B.nbkeys and b > B.keys[i]:
            i += 1
            c += 1
    else:
        c += mystery(B.children[i], a, b)
        while i < B.nbkeys and b > B.keys[i]:
            c += mystery(B.children[i+1], a, b) + 1
            i += 1
    return c
```

1. Let B_1 be the tree in figure 2. For each of the following calls:
   - what is the result?
   - how many calls of `mystery` have been done?

   (a) `mystery(`B_1`, 1, 77)`
   (b) `mystery(`B_1`, 10, 30)`

2. Let `B` be any non-empty B-tree filled with integers, and `a` and `b` two integer values such that `a` $<$ `b`. What does the function `mystery(B, a, b)` calculate?

Appendix

Trees

The (general) trees we work on are the same as the ones in tutorials.

Classical implementation

```python
class Tree:
    def __init__(self, key, children=None):
        self.key = key
        if children is not None:
            self.children = children
        else:
            self.children = []
    @property
    def nbchildren(self):
        return len(self.children)
```

First child - right sibling implementation

```python
class TreeAsBin:
    def __init__(self, key, child=None, sibling=None):
        self.key = key
        self.child = child
        self.sibling = sibling
```

B-Trees

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

```python
class BTree:
    degree = None
    def __init__(self, keys=None, children=None):
        self.keys = keys if keys else []
        self.children = children if children else []
    @property
    def nbkeys(self):
        return len(self.keys)
```

Authorised functions and methods

- `len` on lists.
- `range`
- `append` on lists

`Queue`:

- `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 know what they do).
In any case, the last written function should be the one which answers the question.