Bases de la programmation impérative

Ensimag 1A

1. Entiers récursifs

On se propose ici de stocker des entiers *naturels* comme une liste de bits.

On utilise le type `Cellule` suivant :

```
1  class Cellule:
2      """
3      cellule d'une liste de bits.
4      """
5      # pylint: disable=too-few-public-methods
6      def __init__(self, bit, suivant=None):
7          self.bit = bit
8          self.suivant = suivant
```

Contrairement aux listes vues précédemment, il n'y a pas ici de classe `Liste`. On considère directement la liste des cellules à partir d'une première cellule donnée.

On stocke les bits en partant du bit de poids faible. L'entier 13 (1101 en binaire) sera donc codé par une liste de 4 cellules : une première cellule contenant le bit 1, pointant sur une cellule contenant le bit 0, puis une autre contenant 1 et une dernière contenant 1. Les listes devront être sous forme *canonique*, c'est-à-dire *sans 0 inutiles en queue de liste*. Chaque entier aura donc un codage unique, et 0 sera codé par `None`.

Dans cet exercice, *toutes* les fonctions demandées *doivent être récursives*.

1.1. Dessinez la liste codant 8. Dessinez la liste codant 19.

1.2. Programmez une fonction `valeur(entier_liste)` renvoyant l'entier (l'`int`) codé par la liste de bits.

1.3. Programmez une fonction `construit_liste(entier)` prenant un entier naturel (un `int` ≥ 0) et renvoyant la liste de bits qui le code.

1.4. Programmez l'opérateur de stringification (`__str__`) pour la classe `Cellule`. Cet opérateur devra servir à afficher les bits dans le sens classique de lecture : *du poids fort au poids faible*.

1.5. Programmez une fonction `successeur(entier_liste)` renvoyant la liste codant 1 de plus que l'entier codé par la liste en entrée.

1.6. Programmez une fonction `plus(entier_liste1, entier_liste2)` renvoyant la liste codant la somme des entiers codés par les listes en entrée.

1.7. Programmez une fonction `moins(entier_liste1, entier_liste2)` renvoyant la liste codant la différence des entiers codés par les deux listes en entrée. Cette fonction ne peut pas renvoyer d'entier négatif et doit donc lever une exception si l'entier codé par `entier_liste1` est strictement inférieur à l'entier codé par `entier_liste2`.