Bases de la programmation impérative

Ensimag 1A
Examen final

1 Introduction

On s'intéresse dans ce sujet au développement de listes ordonnées entrelacées d'entiers multiples de 2 et de 3.

Nous vous demandons de compléter le code fourni. **Il est *vital* de vérifier au fur et à mesure du développement que tout fonctionne. Afin de vous forcer la main, nous vous fournissons un script de test qui évalue 15 points de votre note finale. Les 5 points restants seront à la discrétion du correcteur pour noter la clarté de votre code. Nous vous conseillons de lancer ce script le plus souvent possible afin de ne pas avoir de mauvaise surprise en fin d'examen.**

De plus, si *pylint* lève des erreurs ou des avertissements sur votre code vous serez pénalisé.

Nous vous demandons de compléter les différentes méthodes de la classe `Liste` du fichier *liste.py*.

2 Explications

Pour nos listes, chaque `Cellule` est composée d'une `valeur` (un entier multiple de 2 et/ou 3) ainsi que d'un vecteur de 2 cellules suivantes. Ce vecteur respecte les règles suivantes (pour toute cellule) :

- si `cellule.valeur` est impaire alors `cellule.suivants[0]` est `None`
- si `cellule.valeur` n'est pas multiple de 3 alors `cellule.suivants[1]` est `None`
- si `cellule.valeur` est paire alors `cellule.suivants[0]` est une référence vers la prochaine cellule contenant une valeur paire (si elle existe, `None` sinon).
- si `cellule.valeur` est multiple de 3 alors `cellule.suivants[1]` est une référence vers la prochaine cellule contenant une valeur multiple de 3 (si elle existe, `None` sinon).
- pour `index` entre 0 et 1 on a toujours :
  - soit `cellule.suivants[index] is None`
  - soit `cellule.suivants[index].valeur > cellule.valeur`

Toute liste contient toujours une cellule de tête contenant 0.

La figure 1 donnée est un dump de la liste 0, 2, 3, 4, 6, 8, 9 réalisé avec `data_tycat`. Remarquez que chaque cellule multiple de 6 a deux suivants.

3 Démarrage

Nous vous conseillons de passer avant de démarrer quelques minutes pour bien comprendre la structure demandée. N'hésitez pas à faire quelques dessins à la main de listes.

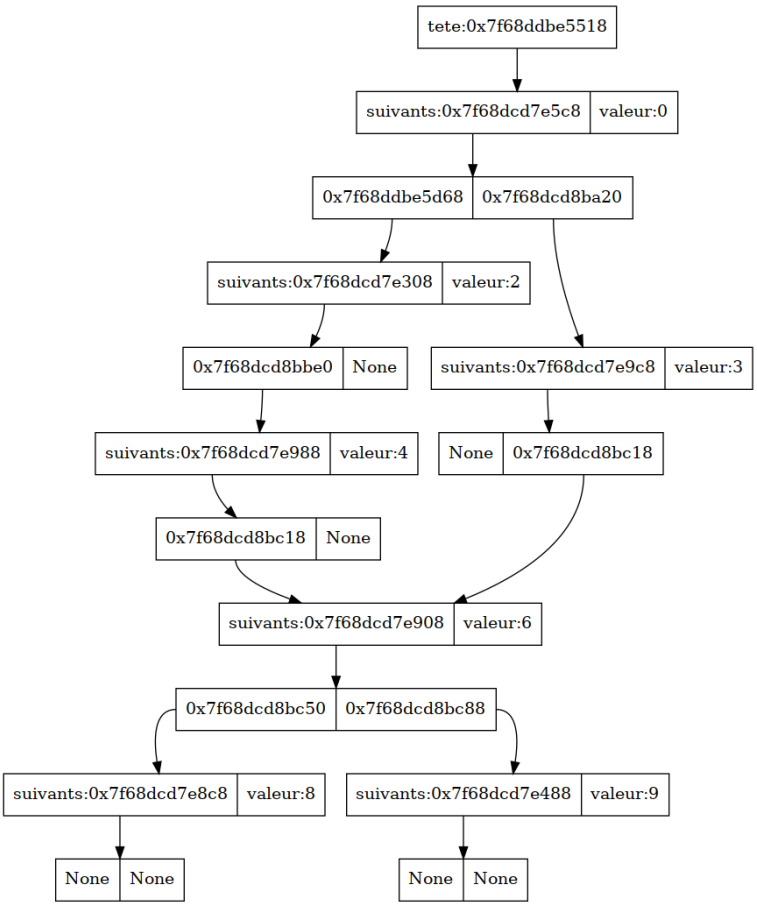

FIGURE 1 – liste contenant 0, 2, 3, 4, 6, 8, 9

Vous pouvez par exemple essayer de dessiner sur papier la `liste_test` puis vérifier que vous avez bien compris avec `data_tycat`.

4 À Implémenter

On vous demande d'implémenter les méthodes suivantes de la classe `Liste` :

- `cellules` prend un index (0 ou 1) et itère sur toutes les cellules paires (0) ou sur toutes les cellules multiples de 3 (1). (1pt)
- `peupler` remplit une liste vide (contenant seulement 0) par toutes les valeurs données. Doit fonctionner en temps linéaire. (3pt)
- `insertion` insère une valeur au bon endroit dans la liste. (3pt)
- `suppression` supprime une valeur de la liste. (2pt)
- `parcours` itère sur toutes les cellules, de la plus petite valeur à la plus grande. On vous demande de tout faire en une seule passe. (5pt)

En plus de ces implémentations, on vous demande comme dernier exercice de lever des exceptions lorsque les pré-conditions listées pour les différentes méthodes ne sont pas respectées. (1pt) Vous pouvez utiliser `assert` ou `raise`, à votre convenance.

Le barème est donné à titre indicatif et pourra être librement modifié par l'équipe enseignante.