Exercices

Exercice 1 * On décide de créer une liste chaînée à l'aide de tuples.

```
# a est la tête de la liste
lst = ("a", ("b", ("c", ("d", ()))))
```

1. Écrire la fonction **longueur(lst: tuple) → int** qui renvoie la taille de la liste.
2. Écrire la fonction **afficher(lst: tuple) → str** qui renvoie une chaîne de caractère de la forme

```
a - b - c - d - fin
```

Exercice 2 ** On souhaite compléter l'interface de la liste chaînée vue en classe, avec une fonctionnalité d'insertion. Pour insérer l'entier 4 à la position 2, il faut procéder en plusieurs étapes:

a. Parcourir (récursivement) la liste tant que la position 2 n'est pas atteinte.
b. Créer un maillon et le faire pointer sur l'élément de rang 2.

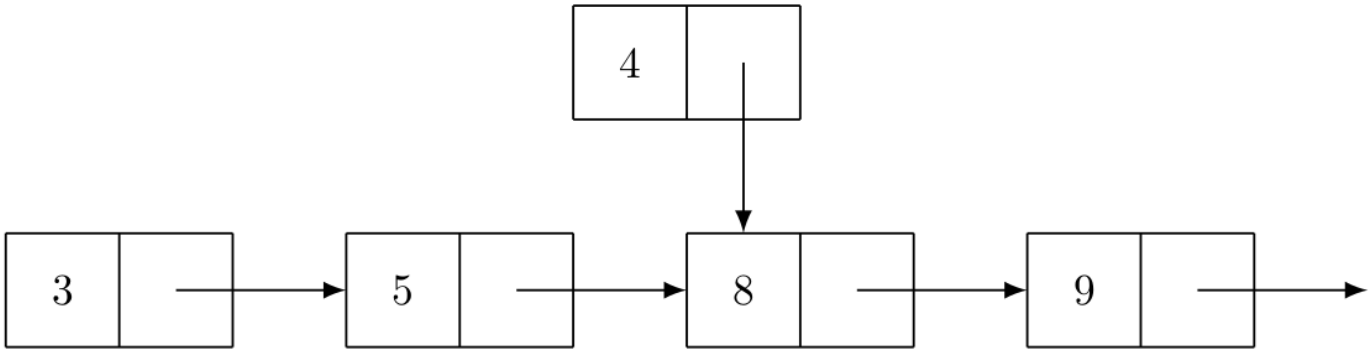

Figure 1: étape 1

c. Faire pointer l'élément de rang 1 sur le nouveau maillon.

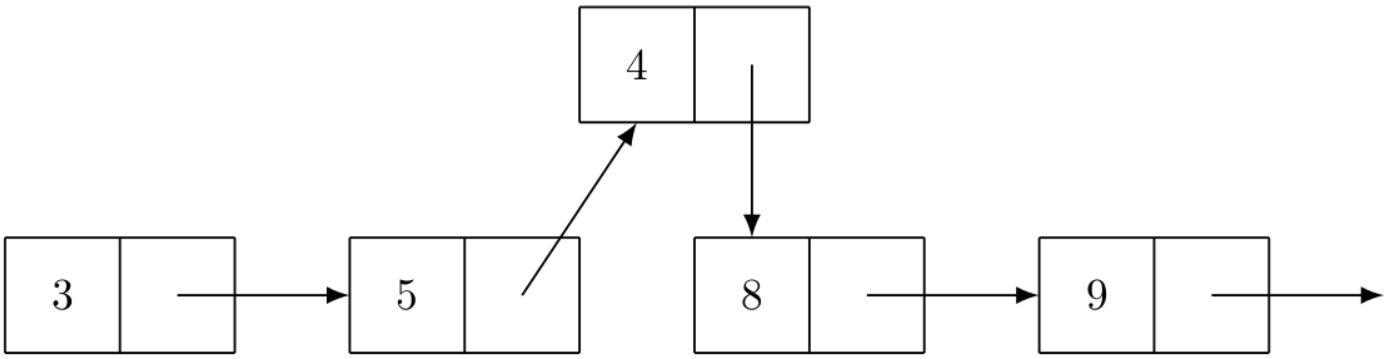

Figure 2: étape 2

1. Écrire la méthode récursive **inserer_rec(self, val: int, n: int, m: object) → None** qui insère l'élément au rang n. Il faudra gérer 4 cas:
   - n = 0: on place l'élément en tête
   - n = 1: la prochaine position est la bonne
   - le suivant est vide: la valeur de n est trop grande, on place l'élément en fin
   - cas général: appel récursif
2. Écrire la méthode **inserer(self, val: int, n: int) → None** qui insère l'élément val au rang n en appelant la méthode précédente.

Exercice 3 **

1. Écrire la méthode impérative **dernier_imp** qui renvoie le dernier élément de la liste. La méthode renverra **-1** si la liste est vide.
2. Proposer une version récursive de la méthode précédente.

Exercice 4 **

1. Proposer un algorithme impératif qui permet de renverser la liste. Exemple: La liste:

- 1 - 2 - 3 - 4 - fin

devient

- 4 - 3 - 2 - 1 - fin

2. Écrire la méthode **renverser** qui implémente cet algorithme.

Exercice 5 **

1. Proposer un algorithme impératif qui permet de dupliquer la liste. Exemple: La liste:

- 1 - 2 - 3 - 4 - fin

devient

- 1 - 1 - 2 - 2 - 3 - 3 - 4 - 4 - fin

2. Écrire la méthode **dupliquer** qui implémente cet algorithme.

Exercice 6 ***

1. Créer un nouveau fichier et importer les classes **Liste, Maillon**.
2. Créer deux listes **l1, l2** et ajouter au moins trois entiers dans chaque liste.
3. Écrire une fonction **concatener(l1: Liste, l2: Liste)** $\rightarrow$ **Liste** qui renverra une **Liste**, concaténation des deux passées en paramètre. Cette fonction contiendra une fonction interne récursive **concatener_rec(tete1: Maillon, tete2: Maillon)** $\rightarrow$ **Maillon** qui implémentera le principe de la figure.

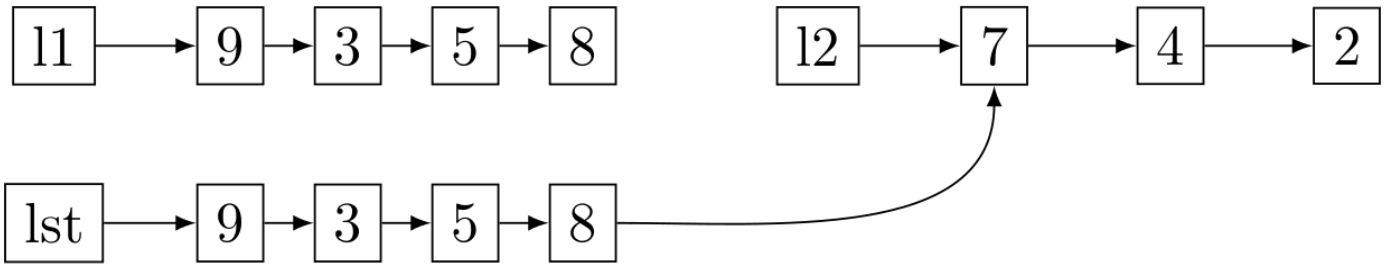

Figure 3: Contaténer

On remarquera en particulier que cette fonction ne copie pas la **Liste** 2.