Interro — Structures abstraites 1 — Correction

Exercice 1 *(1 point)*

Le type abstrait, encore appelé interface, est le cahier des charges d'une structure : elle contient les éléments essentiels (appelés primitives) ainsi que leur nom. Un utilisateur peut donc utiliser l'interface d'un type abstrait pour produire du code, sans se soucier de la mécanique interne.

L'implémentation est la manière d'écrire un type abstrait : un concepteur peut trouver des manières plus ou moins efficaces pour écrire des parties du type abstrait.

Exercice 2 *(3 point)*

On considère une liste chainée L représentée ci-dessous :

a. **`print(L.tete.valeur)`** donne "astérix".

b. **`print(L.tete.suivant.suivant)`** donne le maillon dont la valeur est "idéfix"

c. On va dire que le suivant du maillon idéfix est None ! On connait le maillon "idéfix" grâce à la question b). Donc : **`L.tete.suivant.suivant.suivant = None`**

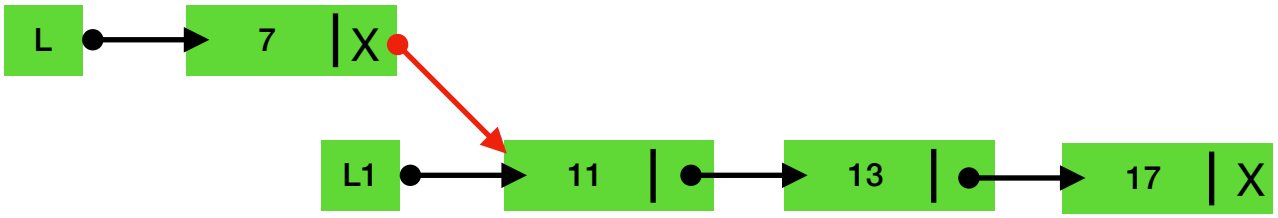
Sans doute plus propre :

```
m = L.tete
while m.suivant.valeur != "pano" :
    m=m.suivant
m.suivant = None
```

Exercice 3 *(3 points)*

a.

b.



c. voir ci-dessus : la modification est représentée par une flèche rouge. On a donc ajouté deux listes : c'est une concaténation.

d. L contient maintenant [7, 88, 13,17] et L1 contient [88, 13, 17]. On a simultanément changé L et L1 ce qui n'était pas du tout notre intention !
Cela arrive en Python lorsque l'on souhaite faire une copie de liste : L=L1 "copie" le pointeur du tableau L vers celui du tableau L1. Si on modifie L1, on va alors aussi modifier L.

Exercice 4 *(3 points)*

Au sein de la classe ListeC, on propose la méthode **mystere** ci-dessous :

```
1  def mystere(self, valeur):
2      if self.tete is None:
3          self.tete = Maillon()
4          self.tete.valeur = valeur
5      else:
6          m = self.tete
7          while m.suivant is not None:
8              m = m.suivant
9          m.suivant = Maillon()
10         m.suivant.valeur=valeur
```

a. On crée une liste chainée L vide.

On appelle ensuite la méthode mystere. La ligne 1 teste si la tete de L est vide. C'est le cas. On crée ensuite un Maillon et on lui donne pour valeur la valeur NSI.

On va donc avoir pour liste L :

b. Dans le cas où la liste chainée est vide, elle rajoute une valeur à la tête en créant le premier maillon.

c. Le code se déplace de maillon en maillon (ligne 7), jusqu'à ce que l'on arrive à un maillon dont le suivant est None. Ce maillon donc le dernier de la liste chainée (il n'a pas de suivant).

Quand on est arrivé au dernier maillon de la liste chainée, on ajoute à la liste un Maillon vide. Puis, on lui donne pour valeur la valeur entrée par l'utilisateur.

La méthode mystere permet donc de faire des ajouts de maillons dans une liste chainée.

Exercice 4 *(2 points)*

Compléter les lignes 2 et 5 du programme ci-dessous de sorte que la fonction `produitDesChiffres(n)` calcule le produit des chiffres composant l'entier positif n.
Par exemple, `produitDesChiffres(243)` doit renvoyer 2 × 4 × 3 = 24.

```
def produitDesChiffres(n):
    if ......:
        return n
    else:
        return n%10 * ..................
```