

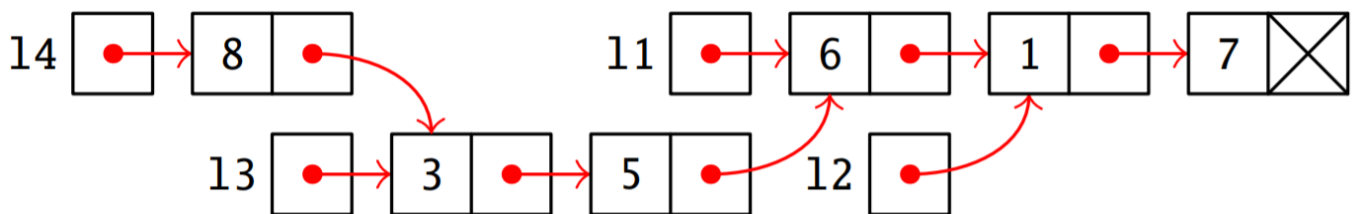
Nom Prénom :

Listes chaînées :

On considère des listes chaînées, avec la liste vide notée `nil` et les fonctions suivantes :

Fonction	Description
<code>tete(liste)</code>	Renvoie la valeur du premier maillon de <code>liste</code> , qui ne doit pas être vide.
<code>queue(liste)</code>	Renvoie la liste sur laquelle pointe le premier maillon de <code>liste</code> , qui ne doit pas être vide.
<code>cons(valeur, liste)</code>	Renvoie une nouvelle liste correspondant à l'ajout de <code>valeur</code> en début de <code>liste</code> .
<code>est_vide(liste)</code>	Renvoie un booléen indiquant si <code>liste</code> est vide ou non.

Le schéma suivant correspond à la représentation en mémoire des listes chaînées `l1`, `l2`, `l3` et `l4`.



Exercice 1 :

Pour chacune des commandes suivantes, indiquer la réponse obtenue.

1. `tete(l2)` : .....
2. `queue(l1)` : .....
3. `est_vide(queue(l1))` : .....
4. `tete(queue(queue(queue(l4))))` : .....
5. `est_vide(queue(queue(l2)))` : .....

Exercice 2 :

L'instruction `l1 = cons(6, cons(1, cons(7, nil)))` définit la liste `l1`. Donner les définitions des listes `l2`, `l3` et `l4` à l'aide des fonctions `cons` et `queue` et en réutilisant les listes déjà définies pour définir les suivantes :

1. `l2` = .....
2. `l3` = .....
3. `l4` = .....

Exercice 3 :

En utilisant les fonctions données ci-dessus, proposer une version itérative pour la fonction `longueur(liste)` qui renvoie la longueur de la liste chaînée `liste` telle que :

```
>>> longueur(nil)
0
>>> longueur(cons(7, cons(0, cons(12, nil))))
3
```

```
def longueur(liste) : # version itérative
```