Lycée Les 3 Sources — **Bourg-Lès-Valence**
NSI T^le — Année **2025-26**

Test n°1

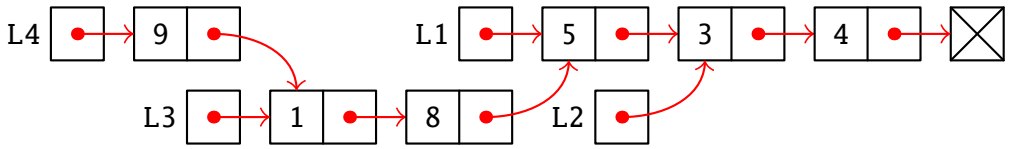
Nom et prénom :

Listes chaînées

Pour toute cette partie, on considère des listes chaînées, avec la liste vide notée `nil` et les fonctions suivantes :

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

Exercice 1 : (4pt) Le schéma suivant correspond une représentation des listes chaînées `L1`, `L2`, `L3` et `L4`.



1) Pour chacune des commandes suivantes, déterminer la réponse obtenue.

```
>>> tete(L2)
3
>>> est_vide(queue(L4))
False
```

```
>>> tete(queue(queue(L4)))
8
>>> est_vide(queue(queue(queue(L1))))
True
```

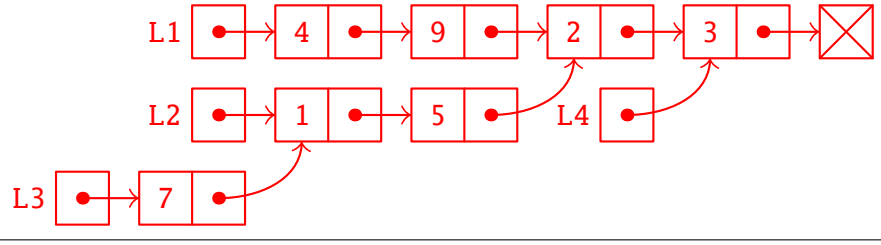
2) Donner les définitions des listes à l'aide des fonctions `cons` et `queue`. Il faut utiliser les listes précédentes pour définir les suivantes.

```
L1 = cons(5, cons(3, cons(4, nil)))
L2 = queue(L1)
L3 = cons(1, cons(8, L1))
L4 = cons(9, L3)
```

Exercice 2 : (2pt) Représenter l'état de la mémoire après les instructions suivantes. Vous pouvez vous inspirer du schéma de l'exercice précédent.

```
L1 = cons(4, cons(9, cons(2, cons(3, nil))))
L2 = cons(1, cons(5, queue(queue(L1))))
L3 = cons(7, L2)
L4 = queue(queue(queue(L2)))
```

Solution :

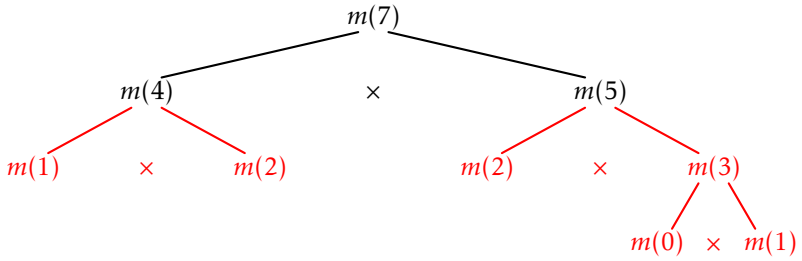


Récursivité

Exercice 3 : (2pt) On considère la fonction suivante :

$$m(n) = \begin{cases} 2 & \text{si } n = 0 \\ 3 & \text{si } n = 1 \\ 5 & \text{si } n = 2 \\ m(n-3) \times m(n-2) & \text{si } n > 2 \end{cases}$$

1) Voici le début de l'arbre d'appel de $m(7)$ qui représente les appels récursifs et les opérations à faire pour obtenir le résultat. Compléter cet arbre.



2) Calculer la valeur de $m(7)$ en indiquant les valeurs intermédiaires.
Solution : On fait les calculs intermédiaires :

$m(0) = 2$
$m(1) = 3$
$m(2) = 5$
$m(3) = m(0) \times m(1) = 6$
$m(4) = m(1) \times m(2) = 15$
$m(5) = m(2) \times m(3) = 30$
$m(6) = m(3) \times m(4) = 90$
$m(7) = m(4) \times m(5) = 450$

Exercice 4 : (1pt) On considère la fonction ci-contre.

```
def mystere(n):
    if n == 0:
        return 1
    else:
        return n * mystere(n-1)
```

1) Quel est le résultat de `mystere(5)` ? 120
2) Décrire, en français, ce que fait cette fonction.
Solution : Cette fonction calcule le produit de tous les entiers de 1 à n, c'est-à-dire la factorielle de n.

Exercice 5 : (2pt) On souhaite écrire une fonction récursive `compte(val, liste)` qui compte le nombre d'occurrences de `val` dans la **liste chaînée** `liste`.
Compléter le code de la fonction `compte(val, liste)` en Python en utilisant les fonctions sur les listes chaînées.

```
def compte(val, liste):
    if est_vide(liste):
        return 0
    elif val == tete(liste):
        return 1 + compte(val, queue(liste))
    else:
        return compte(val, queue(liste))
```

Lycée Les 3 Sources — **Bourg-Lès-Valence**
NSI T^le — Année **2025-26**

Test n°1

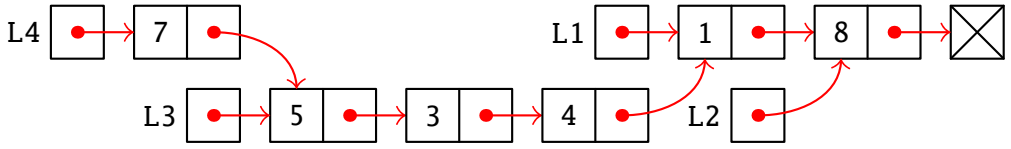
Nom et prénom :

Listes chaînées

Pour toute cette partie, on considère des listes chaînées, avec la liste vide notée `nil` et les fonctions suivantes :

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

Exercice 1 : (4pt) Le schéma suivant correspond une représentation des listes chaînées L1, L2, L3 et L4.



1) Pour chacune des commandes suivantes, déterminer la réponse obtenue.

```
>>> tete(L2)
8
>>> est_vide(queue(L2))
True
```

```
>>> tete(queue(queue(L4)))
3
>>> est_vide(queue(queue(queue(L3))))
False
```

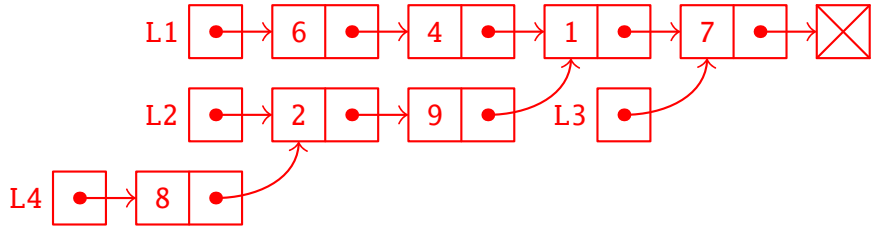
2) Donner les définitions des listes à l'aide des fonctions `cons` et `queue`. Il faut utiliser les listes précédentes pour définir les suivantes.

```
L1 = cons(1, cons(8, nil))
L2 = queue(L1)
L3 = cons(5, cons(3, cons(4, L1)))
L4 = cons(7, L3)
```

Exercice 2 : (2pt) Représenter l'état de la mémoire après les instructions suivantes. Vous pouvez vous inspirer du schéma de l'exercice précédent.

```
L1 = cons(6, cons(4, cons(1, cons(7, nil))))
L2 = cons(2, cons(9, queue(queue(L1))))
L3 = queue(queue(queue(L2)))
L4 = cons(8, L2)
```

Solution :

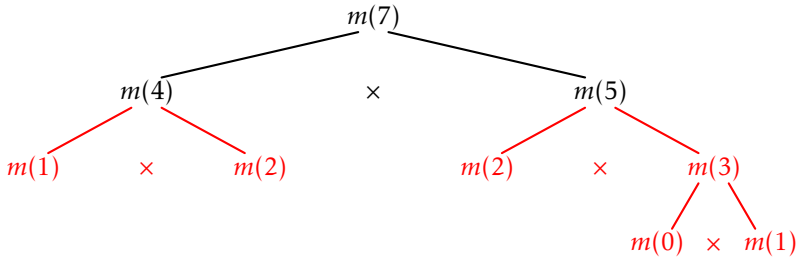


Récursivité

Exercice 3 : (2pt) On considère la fonction suivante :

$$m(n) = \begin{cases} 2 & \text{si } n = 0 \\ 4 & \text{si } n = 1 \\ 5 & \text{si } n = 2 \\ m(n-3) \times m(n-2) & \text{si } n > 2 \end{cases}$$

1) Voici le début de l'arbre d'appel de $m(7)$ qui représente les appels récursifs et les opérations à faire pour obtenir le résultat. Compléter cet arbre.



2) Calculer la valeur de $m(7)$ en indiquant les valeurs intermédiaires.
Solution : On fait les calculs intermédiaires :

$m(0) = 2$
$m(1) = 4$
$m(2) = 5$
$m(3) = m(0) \times m(1) = 8$
$m(4) = m(1) \times m(2) = 20$
$m(5) = m(2) \times m(3) = 40$
$m(6) = m(3) \times m(4) = 160$
$m(7) = m(4) \times m(5) = 800$

Exercice 4 : (1pt) On considère la fonction ci-contre.

```
def mystere(n):
    if n == 0:
        return 1
    else:
        return n * mystere(n-1)
```

1) Quel est le résultat de `mystere(4)` ? 24
2) Décrire, en français, ce que fait cette fonction.
Solution : Cette fonction calcule le produit de tous les entiers de 1 à n, c'est-à-dire la factorielle de n.

Exercice 5 : (2pt) On souhaite écrire une fonction récursive `compte(val, liste)` qui compte le nombre d'occurrences de `val` dans la **liste chaînée** `liste`.
Compléter le code de la fonction `compte(val, liste)` en Python en utilisant les fonctions sur les listes chaînées.

```
def compte(val, liste):
    if est_vide(liste):
        return 0
    elif val == tete(liste):
        return 1 + compte(val, queue(liste))
    else:
        return compte(val, queue(liste))
```