Exercice 1 :

On propose, en pseudo code, la fonction suivante mystere(L,i)
où L est une liste chaînée et i une clé de cette liste :

```
1. def mystere(L,i) :
2.       x=tete[L]
3.       si cle[x]==i :
4.          tete(L)=succ[x]
5.       sinon :
6.          tant que cle[x]!=i et pointeVers[succ[x]]!=NIL :
7.                y=x
8.                x=succ[x]
9.          pointeVers[succ[y]]=pointeVers[succ[x]]
10.      return L
```

1) En utilisant les tableaux donnés ci-dessous, faire la trace d'exécution de mystere(L,cle[]), puis de mystere(L,cle[a]), où L est la liste schématisée par le dessin de droite.
L contient donc dans l'ordre les éléments {.

i	
x	
succ[x]	
test : cle[x]==i ?	
tete(L)	
pointeVers[succ[x]]	
boucle tant que	
cle[x]!=i et	
pointeVers[succ[x]]!=NI	
L	
y	
pointeVers[succ[y]]	

i	
x	
succ[x]	
test : cle[x]==i ?	
tete(L)	
pointeVers[succ[x]]	
boucle tant que	
cle[x]!=i et	
pointeVers[succ[x]]!=NIL	
y	
pointeVers[succ[y]]	

2) Donner un nom “parlant“ à cette fonction mystère.

Exercice 2 : Écrire en pseudo code les fonctions suivantes :

TD/TP : Introduction aux structures de
données et étude des listes chaînées

Structures de données
Séquence 2-1

2
longueur(L) : qui prend en paramètres la liste L et qui renvoie le nombre d'éléments de la liste.
insererEnDebut(L,x) : qui prend en paramètre l'élément x et la liste L et renvoie la liste avec
l’élément x qui est inséré à la tête de la liste.
inserer(L,a,e) : qui prend en paramètres la liste L, les éléments a et e et qui renvoie la liste avec
l'élément e inséré après l'élément a de la liste.
Rappel : un élément d’une liste est un couple de la forme (clé[x],succ[x]).

Exercice 3 : Implémentation minimale d'une liste chainée
Il n'existe pas en python de type représentant les listes chaînées. On peut donc implémenter cette
nouvelle structure à l'aide de la POO.
On commencera ici par une implémentation rudimentaire à l'aide d'une classe Maillon. On considèrera
alors une liste comme un maillon de maillons
La liste chaînée : 3 →5 →1 est créée par : Maillon(3, Maillon(5, Maillon(1,None)))
Taper le programme suivant dans un éditeur python.

```
class Maillon :
def __init__(self, valeur, suivant):
self.valeur=valeur
self.suivant= suivant

lst =Maillon(3, Maillon(5, Maillon(1,None))) # crée la liste chaînée : 3 →5 →1
```

Taper les instructions permettant de répondre aux questions suivantes.
1) Comment accéder à l'élément 3 ?
2) Comment accéder à l'élément 5 ?
3) Comment accéder à l'élément 1 ?

TD/TP : Introduction aux structures de
données et étude des listes chaînées

Structures de données
Séquence 2-1

Exercice 4 (pour les plus avancés) : implémentation d'une liste chaînée en python sur PC
On peut maintenant implémenter une liste chainée à l'aide de la POO en utilisant la classe Maillon et
une classe Liste_chaine
1) Créer une classe Maillon qui possède comme attribut une valeur (la valeur de l'élément courant) et
un suivant (la valeur du suivant s'il existe, None sinon). Par défaut, un maillon vide renvoie None.
La classe Maillon contiendra :
Le constructeur : __ init __ (self, valeur, suivant)
La méthode pour l'affichage : __str__(self)
Testez votre code en tapant dans le corps du programme :

```
MonMaillon = Maillon()
print(MonMaillon)
MonMaillon = Maillon(5)
print(MonMaillon)
MonMaillon=Maillon(5,13)
print(MonMaillon)
```

2) Créer une classe Liste_chaine qui possède comme attribut une valeur tete représentant la liste
des premiers éléments. Une liste vide renvoie None.
La classe Liste_chaine contiendra :
Le constructeur : __ init __ (self, tete) qui utilise la classe Maillon.
La méthode pour l'affichage : __str__(self)
La méthode ajout_debut(self,element) qui ajoute un élément en début de liste.
La méthode ajout_fin (self,val) qui ajoute un élément en fin de liste.
Testez votre code en tapant dans le corps du programme :

```
MaListe = Liste_chaine()
print(MaListe)
MaListe. ajout_fin(4)
print(MaListe)
MaListe. ajout_fin(5)
print(MaListe)
MaListe. ajout_debut(3)
print(MaListe)
```

3) Écrire les méthodes suivantes dans la classe Liste_chaine :
La méthode element_tete(self) qui renvoie la valeur du premier élément de la liste.
La méthode supprime_premier(self) qui renvoie la liste sans le premier l'élément.
En dehors de la classe, écrire une fonction récursive longueur(liste) qui renvoie le nombre d'éléments de la liste. Tester vos fonctions.