

Algorithmes sur les listes chaînées

Pour cette feuille, on dispose de l'interface suivantes pour les listes chaînées. On pourra noter nil la liste vide.

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 : Écrire le code de la fonction **itérative** longueur qui prend en paramètre une liste chaînée et qui renvoie sa longueur.

```
def longueur(liste):  
    .....  
    while .....:  
        .....  
        liste = .....  
    return .....
```

EXERCICE 2 : Écrire le code de la fonction **réursive** longueur qui prend en paramètre une liste chaînée et qui renvoie sa longueur.

```
def longueur(liste):  
    if .....:  
        return .....  
    else:  
        return .....
```

EXERCICE 3 : Écrire le code de la fonction **itérative** appartient qui prend en paramètres une valeur val et une liste chaînée, et qui renvoie **True** si val est dans la liste et **False** sinon.

```
def appartient(val, liste):  
    while .....:  
        if .....:  
            return .....  
        .....  
    return .....
```

EXERCICE 4 : Écrire le code de la fonction **réursive** appartient qui prend en paramètres une valeur val et une liste chaînée, et qui renvoie **True** si val est dans la liste et **False** sinon.

```
def appartient(val, liste):  
    if .....:  
        return .....  
    elif .....:  
        return .....  
    else:  
        return .....
```

EXERCICE 5 : Écrire le code de la fonction **réursive** `nb_occ` qui prend en paramètres une valeur `val` et une liste chaînée, et qui renvoie le nombre d'occurrences de `val` dans la liste.

```
def nb_occ(val, liste):
```

EXERCICE 6 : Écrire le code de la fonction **réursive** `somme` qui prend en paramètre une liste chaînée contenant des nombres et qui renvoie la somme des valeurs contenues dans la liste.

```
def somme(liste):
```

EXERCICE 7 : Écrire les versions **itérative** et **réursive** de la fonction `n_ieme` qui prend en paramètres un entier naturel `i` et une liste chaînée, et qui renvoie l'élément d'indice `i` de la liste. On suppose qu'il y a au moins `i+1` éléments dans la liste.

```
def n_ieme(i, liste):
```

```
def n_ieme(i, liste):
```

EXERCICE 8 : Écrire le code de la fonction **itérative** `renverse` qui prend en paramètre liste chaînée `liste` et qui renvoie une nouvelle liste chaînée avec les éléments de `liste` dans l'ordre inverse.

```
def renverse(liste):
```

EXERCICE 9 : Écrire le code de la fonction **réursive** `concat` qui prend en paramètres deux listes chaînées `liste1` et `liste2`, et qui renvoie la concaténation des deux listes. Les maillons de `liste1` sont dupliqués mais on garde ceux de `liste2`.

```
def concat(liste1, liste2):
```