```python
class Maillon:
    """
    Crée un maillon de la liste chaînée
    """
    def __init__(self, val: int, suiv: object)->None:
        self.valeur = val
        self.suivant = suiv

class Liste:
    """
    Crée une liste chaînée
    """
    def __init__(self):
        self.tete: object = None

    def est_vide(self)->bool:
        return self.tete is None

    def ajoute(self, val: int)->None:
        self.tete = Maillon(val, self.tete)

    def __len__(self)->int:
        maillon = self.tete
        taille = 0
        while not(maillon == None):
            maillon = maillon.suivant
            taille += 1
        return taille

    def taille_rec(self, maillon: object)->int:
        """
        méthode interne pour calculer la taille de la chaîne
        """
        if maillon is None:
            return 0
        else:
            return 1 + self.taille_rec(maillon.suivant)

    def taille(self)->int:
        """
        appel principal de la méthode récursive pour mesurer
        la taille de la chaîne
        """
        return self.taille_rec(self.tete)

    def __getitem__(self, n: int)->object:
        """
        renvoie l'élément de rang n. Les indices commencent à0.
        """
        if self.tete is None or n < 0:
            raise IndexError("indice invalide")
```


Christophe Viroulaud

```
52
53          maillon = self.tete
54          i = 0
55          while not(i == n):
56              maillon = maillon.suivant
57              if maillon is None:
58                  raise IndexError("indice invalide")
59              i += 1
60
61          return maillon.valeur
62
63      def get_element_rec(self, n: int, maillon: object)->int:
64          """
65          méthode interne pour renvoyer le n-ième élément.
66          """
67          if maillon is None or n < 0:
68              raise IndexError("indice invalide")
69          if n == 0:
70              return maillon.valeur
71          else:
72              return self.get_element_rec(n-1, maillon.suivant)
73
74      def get_element(self, n: int)->int:
75          """
76          appel principal de la méthode récursive pour renvoyer le n
              -ième élément
77          """
78          return self.get_element_rec(n, self.tete)
79
80  lst = Liste()
81  lst.ajoute(8)
82  lst.ajoute(5)
83  lst.ajoute(3)
84  lst.ajoute(9)
85  print(len(lst))
86  print(lst.taille())
87  print(lst[3])
88  print(lst.get_element(3))
```


Christophe Viroulaud