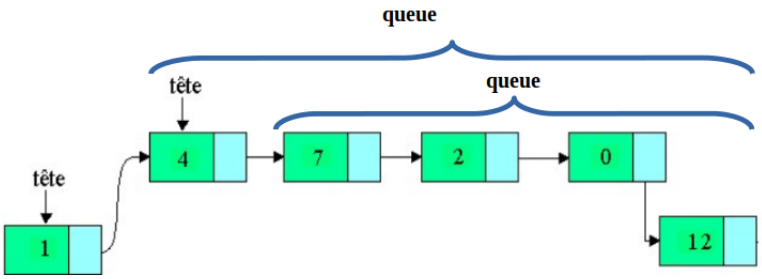
ExerciceType Bac **Épreuve écrite**
NSI *Terminale* Année .....

Exercice 3

POO - Listes - Récursivité

Le principe de la liste chaînée
Le schéma ci-dessous illustre le principe d'une liste chaînée : chaque cellule est constitué d'une valeur (la tête) et d'une queue qui fait référence à la cellule suivante.



On a choisit de représenter une liste à l'aide des deux classes ci-dessous :

```
class Cellule:
    def __init__(self, tete = None, queue = None):
        self.tete = tete
        self.queue = queue

class Liste:
    def __init__(self, c = None):
        self.cellule = c

    # renvoie True si la liste est vide
    def est_vide(self):
        return self.cellule is None

    # ajoute un élément en fin de liste
    def ajouter(self,val):
        if self.est_vide():# on la crée si elle est vide
            self.cellule = Cellule(val)
        else:
            tmp = self.cellule  # on se place en tête de liste
            while tmp.queue is not None:  # on parcourt la liste
                tmp = tmp.queue
            tmp.queue = Cellule(val)  # on ajoute une nouvelle cellule
```

L'objectif de cet exercice est d'écrire une fonction `tri_rapide` qui trie une liste implémentée à partir de la classe `Liste` .

Question 1:

1. Quelle instruction permet de créer la liste vide `liste1`?
2. Que contient la liste1 après exécution de ce programme?

```
t=(1,2,7,2,0,12)
for valeur in t:
    liste1.ajouter(valeur)
```

3. On suppose que l'on également accès à la fonction **`len`**`(liste1)` qui renvoie la longueur de la liste.
   Quel affichage produit alors ce programme?

```
tmp=liste1.cellule
for i in range(len(liste1)):
    print(tmp.tete,end=' ')
    tmp=tmp.queue
```

Voici une méthode de la classe liste qui est sensée supprimer le premier élément de la liste.

```
    # supprime le premier élément de la liste et le renvoie
    def supprimer_premier(self):
        if self.est_vide() == ???:
            x = ???
            self.cellule = ???
            return ???
```

Question 2:

Recopier et compléter la méthode ci-dessus en remplaçant les??? par les bonnes instructions.

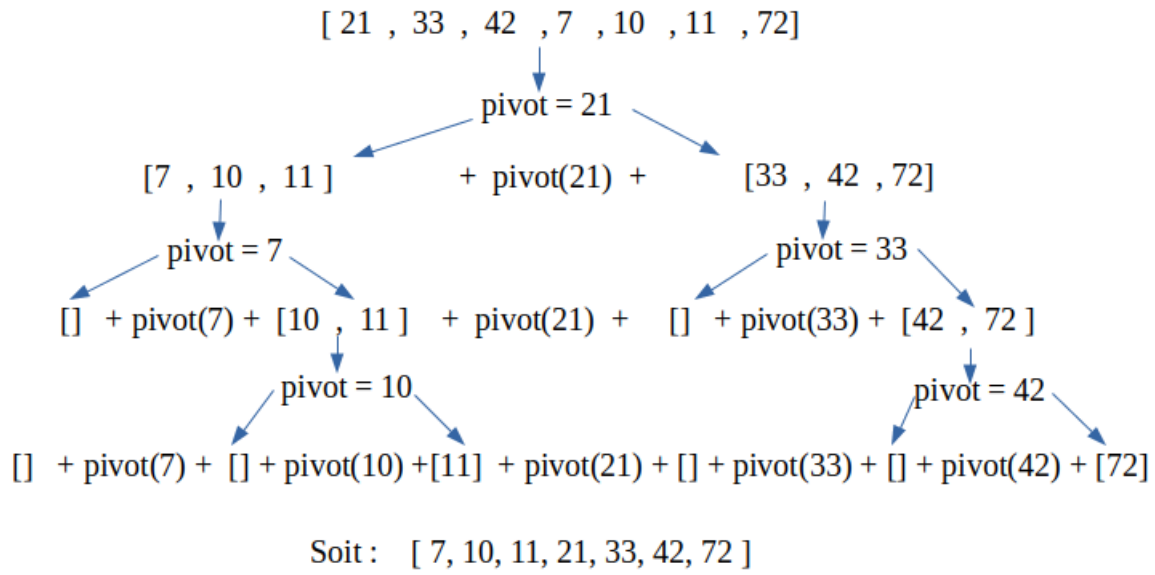
On suppose maintenant que la liste lst1 contient les valeurs suivantes:
28 49 39 22 0 16 5 37 29 34 47 26 15 10 38

Question 3:

Que contiennent les listes `lst2` et `lst3` après exécution de ce programme?

```
 def couper(lst,pivot):
    l1=Liste()
    l2=Liste()
    tmp = lst.cellule
    while tmp !=None:
        if tmp.tete <= pivot:
            l1.ajouter(tmp.tete)
        else:
            l2.ajouter(tmp.tete)
        tmp = tmp.queue
    return l1,l2
pivot = lst1.supprimer_premier()
lst2,lst3=couper(lst1,pivot)
```

On souhaite maintenant trier la liste en appliquant la méthode suivante réalisée sur un exemple:



On suppose que l'on a accès à une fonction `concat(lst1,el,lst2)` qui prend en paramètre deux listes et une valeur et qui renvoie `lst1 + el + lst2`, c'est à dire une liste constituée des éléments de lst1, de el et des éléments de lst2 dans cet ordre.

? Question 4:

En utilisant les fonctions des questions 2 et 3 et de la fonction `concat`, écrire en Python une fonction `tri_rapide(lst)` récursive qui prend en paramètre une liste (définie à partir de la classe `Liste`) et qui renvoie une liste contenant les éléments triés dans l'ordre croissant de la liste passée en paramètre.