

Exercice 1

Thème abordé : structures de données : les piles

On cherche à obtenir un mélange d'une liste comportant un nombre **pair** d'éléments. Dans cet exercice, on notera N le nombre d'éléments de la liste à mélanger.

La méthode de mélange utilisée dans cette partie est inspirée d'un mélange de jeux de cartes :

- On sépare la liste en deux piles :
 - ⇒ à gauche, la première pile contient les $N/2$ premiers éléments de la liste ;
 - ⇒ à droite, la deuxième pile contient les $N/2$ derniers éléments de la liste.
- On crée une liste vide.
- On prend alors le sommet de la pile de gauche et on le met en début de liste.
- On prend ensuite le sommet de la pile de droite que l'on ajoute à la liste et ainsi de suite jusqu'à ce que les piles soient vides.

Par exemple, si on applique cette méthode de mélange à la liste `['V', 'D', 'R', '3', '7', '10']`, on obtient pour le partage de la liste en 2 piles :

Pile gauche	Pile droite
'R'	'10'
'D'	'7'
'V'	'3'

La nouvelle liste à la fin du mélange sera donc `['R', '10', 'D', '7', 'V', '3']`.

1. Que devient la liste `['7', '8', '9', '10', 'V', 'D', 'R', 'A']` si on lui applique cette méthode de mélange ?

On considère que l'on dispose de la structure de données de type pile, munie des seules instructions suivantes :

- `p = Pile()` : crée une pile vide nommée `p`
- `p.est_vide()` : renvoie Vrai si la liste est vide, Faux sinon
- `p.empiler(e)` : ajoute l'élément `e` dans la pile
- `e = p.depiler()` : retire le dernier élément ajouté dans la pile et le retourne (et l'affecte à la variable `e`)
- `p2 = p.copier()` : renvoie une copie de la pile `p` sans modifier la pile `p` et l'affecte à une nouvelle pile `p2`

2. Recopier et compléter le code de la fonction suivante qui transforme une liste en pile.

```
def liste_vers_pile(L):
    '''prend en paramètre une liste et renvoie une
    pile'''
    N = len(L)
    p_temp = Pile()
    for i in range(N):
        .....
    return .....
```

3. On considère la fonction suivante qui partage une liste en deux piles. Lors de sa mise au point et pour aider au débogage, des appels à la fonction `affichage_pile` ont été insérés. La fonction `affichage_pile(p)` affiche la pile `p` à l'écran verticalement sous la forme suivante :

dernier élément empilé
...
...
premier élément empilé

```
def partage(L):
    N = len(L)
    p_gauche = Pile()
    p_droite = Pile()
    for i in range(N/2):
        p_gauche.empile(L[i])
    for i in range(N/2,N):
        p_droite.empile(L[i])
    affichage_pile(p_gauche)
    affichage_pile(p_droite)
    return p_gauche, p_droite
```

Quels affichages obtient-on à l'écran lors de l'exécution de l'instruction :
`partage([1,2,3,4,5,6])` ?

4.

4.a Dans un cas général et en vous appuyant sur une séquence de schémas, **expliquer** en quelques lignes comment fusionner deux piles `p_gauche` et `p_droite` pour former une liste `L` en alternant un à un les éléments de la pile `p_gauche` et de la pile `p_droite`.

4.b. **Écrire** une fonction `fusion(p1,p2)` qui renvoie une liste construite à partir des deux piles `p1` et `p2`.

5. **Compléter** la dernière ligne du code de la fonction `affichage_pile` pour qu'elle fonctionne de manière récursive.

```
def affichage_pile(p):
    p_temp = p.copier()
    if p_temp.est_vide():
        print('____')
    else:
        elt = p_temp.depiler()
        print('| ', elt, ' |')
        ... # ligne à compléter
```