On cherche ici à mettre en place des algorithmes qui permettent de modifier l'ordre des informations contenues dans une file. On considère pour cela les structures de données abstraites de Pile et File définies par leurs fonctions primitives suivantes :

Pile :

- `creer_pile_vide()` renvoie une pile vide ;
- `est_pile_vide(p)` renvoie `True` si la pile `p` est vide, `False` sinon ;
- `empiler(p, element)` ajoute element au sommet de la pile `p` ;
- `depiler(p)` renvoie l'élément se situant au sommet de la pile `p` en le retirant de la pile `p` ;
- `sommet(p)` renvoie l'élément se situant au sommet de la pile `p` **sans le retirer de la pile**.

File :

- `creer_file_vide()` renvoie une file vide ;
- `est_file_vide(f)` renvoie `True` si la file `f` est vide, `False` sinon ;
- `enfiler(f, element)` ajoute element dans la file `f` ;
- `defiler(f)` renvoie l'élément à la tête de la file `f` en le retirant de la file `f` .

On représentera les files par des éléments en ligne, l'élément de droite étant la tête de la file et l'élément de gauche étant la queue de la file.
On représentera les piles en colonnes, le sommet de la pile étant le haut de la colonne.

La file suivante est appelée `f` : **4 3 8 2 1**

La pile suivante est appelée `p` :
5
8
6
2

⚠️ **Les quatre questions suivantes sont indépendantes. Pour chaque question, on repartira de la pile `p` et de la file `f` initiales présentées ci-dessus.**

1. Représenter la file `f` après l'exécution du code suivant : `enfiler(f, defiler(f))`

2. Représenter la pile `p` après l'exécution du code suivant : `empiler(p, depiler(p))`

3. Représenter la pile `p` et la file `f` après l'exécution du code suivant :

```
for i in range(2):
    enfiler(f, depiler(p))
```

4. Représenter la pile `p` et la file `f` après l'exécution du code suivant :

```
for i in range(2):
    empiler(p, defiler(f))
```

La fonction `mystere` prend une file en argument, modifie cette file et renvoie une pile :

```
def mystere(f):
    p = creer_pile_vide()
    while not est_file_vide(f):
        empiler(p, defiler(f))
    while not est_pile_vide(p):
        enfiler(f, depiler(p))
    return p
```

5. Préciser l'état de la variable `f` après **chaque** boucle de la fonction `mystere` appliquée à la file **1 2 3 4** et indiquer le contenu de la pile renvoyée par la fonction.