✅ EXERCICE 1 — Piles, files

On rappelle les structures initiales :

- **File f** : `4 3 8 2 1`
  (4 à gauche = queue, 1 à droite = tête)
- **Pile p** :

```
5   ← sommet
8
6
2
```

---

1. Manipulations Pile/File

a. `enfiler(f, defiler(f))`

- `defiler(f)` retire **1** (tête)
- `enfiler(f, 1)` place **1** en queue

Nouvelle file :

➡️ **4 3 8 2 1 → 3 8 2 1 1**
(mais attention ! 4 était en queue → le defiler retire 1 → nouvelle file : `4 3 8 2` → on enfiler 1 → `4 3 8 2 1`)

Correction finale :
👉 **La file reste identique : 4 3 8 2 1**

En réalité :
`defiler` retire `1`
File devient : **4 3 8 2**
Puis `enfiler(1)`
→ **4 3 8 2 1**

✔️ **Réponse** : `4 3 8 2 1`

---

b. `empiler(p, depiler(p))`

La pile :

```
5   ← sommet
8
6
```

```
2
```

- `depiler(p)` retire **5**
- `empiler(p, 5)` remet 5 au sommet

✔ La pile est **inchangée**.

---

c.

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

Déroulement :

Itération 1 :

- depiler(p) → 5
- enfiler(f, 5)

Itération 2 :

Nouvelle pile :

```
8 ← sommet
6
2
```

- depiler(p) → 8
- enfiler(f, 8)

Résultats :

✔ **Pile p** :

```
6 ← sommet
2
```

✔ **File f** :

Initiale : `4 3 8 2 1` → devient :
→ après avoir ajouté 5 : `4 3 8 2 1 5`
→ après avoir ajouté 8 : `4 3 8 2 1 5 8`

---

d.

```
for i in range(2):
```

```
    empiler(p, defiler(f))
```

File initiale : `4 3 8 2 1`

Itération 1 :

- defiler(f) → 1
- empiler(p, 1)

Itération 2 :

Nouvelle file : `4 3 8 2`

- defiler(f) → 2
- empiler(p, 2)

Résultats :

✔ **Pile p** :

```
2 ← nouvel élément
1
5
8
6
2
```

✔ **File f** : `4 3 8`

---

2. Fonction mystere

```
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
```

Pour une file initiale : **f = 4 3 8 2 1**

Après la première boucle :

La file est vidée :

➡️ f = vide

La pile contient les éléments **dans l'ordre inverse** :

```
1
2
8
3
4
```

(1 au sommet)

Après la deuxième boucle :

On dépile p → on ré-enfile dans f ⇒ la file redevient **inversée** :

➡️ **f = 1 2 8 3 4**

Valeur renvoyée par mystere

La pile finale est **vide** car elle a été entièrement dépilée.

✔️ **f devient la file renversée**
✔️ **p renvoyée = pile vide**

---

3. Fonction knuth(f)

On applique à la file : **2, 1, 3**

Je donne le tableau corrigé (concis) :

Étape	File f	Pile p
Initial	2,1,3	vide
1	1,3	2
2	3	2,1 (1 < 2 → retour en file)
3	3,2	1
4	2	1,3
5	vide	1,3,2 (puis tout sera vidé dans f)

Résultat final après la dernière boucle :

➡️ **f = 1,3,2**

b. Que fait l'algorithme ?

C'est **l'algorithme de tri de Knuth par empilement** :

👉 **Il trie la file dans l'ordre croissant**.

✅ EXERCICE 2 — Programmation objet

1.a. Fonction donnePremierIndiceLibre

```
def donnePremierIndiceLibre(Mousse):
    i = 0
    while i < 6 and Mousse[i] != None:
        i = i + 1
    return i
```

1.b. Fonction placeBulle(B)

```
def placeBulle(B):
    i = donnePremierIndiceLibre(Mousse)
    if i < 6:
        Mousse[i] = B
```

---

2. Fonction bullesEnContact

```
def bullesEnContact(B1, B2):
    return distanceEntreBulles(B1, B2) <= B1.rayon + B2.rayon
```

---

3. Fonction collision

Compléter :

```
def collision(indPetite, indGrosse, Mousse):
    surfPetite = pi * Mousse[indPetite].rayon**2
    surfGrosse = pi * Mousse[indGrosse].rayon**2
    surfGrosseApresCollision = surfPetite + surfGrosse

    rayonGrosseApresCollision = sqrt(surfGrosseApresCollision / pi)

    Mousse[indGrosse].rayon = rayonGrosseApresCollision

    Mousse[indGrosse].dirx = Mousse[indGrosse].dirx / 2
    Mousse[indGrosse].diry = Mousse[indGrosse].diry / 2

    Mousse[indPetite] = None
```

✔ Rayon mis à jour
✔ Vitesse divisée par 2
✔ Petite bulle supprimée
