Exercice 1 : Il est possible de construire une pile à partir d'un simple tableau. On note `p` une pile vide.

```
1 p = []
```

1. Écrire les fonctions implémentant l'interface de la pile.
   - `est_vide(p: list)` → `bool`
   - `empiler(p: list, e: int)` → `None`
   - `depiler(p: list)` → `int`
2. Sur le même modèle, construire une file.
3. Quel est l'inconvénient d'utiliser un tableau pour construire ce type de structure ?

Exercice 2 : Il est possible de créer une file avec deux piles en appliquant l'algorithme suivant :

- Enfiler un élément dans la pile gauche.
- Défiler un élément dans la pile droite. Si la pile est vide, dépiler la pile gauche dans la pile droite.

1. Vérifier sur le papier le fonctionnement de la file.
2. Reprendre la classe `Pile` construite en cours et implémenter la classe **File2** qui respecte l'algorithme précédant. Elle contiendra des entiers.

Exercice 3 : Flavius Josèphe était un historien juif du premier siècle. Il participa aux révoltes contre les Romains et fut assiégé avec quarante de ses compatriotes dans la forteresse de Jotapata en 67. Les extrémistes du groupe persuadèrent l'ensemble de se tuer pour ne pas tomber aux mains des Romains. Ne partageant pas ce point de vue mais n'osant s'opposer au groupe, Josèphe proposa que l'on se mette en cercle et que chaque troisième personne soit tuée, la dernière devant se suicider.
En utilisant une *file* trouver la position à choisir parmi les 41 soldats pour être le dernier éliminé.

Exercice 4 : Un EDI se charge de vérifier si le code écrit par le programmeur est correctement parenthésée, c'est à dire si à chaque parenthèse ouvrante correspond une parenthèse fermante.

1. En utilisant une pile, trouver sur papier un algorithme permettant de vérifier le parenthésage d'une chaîne de caractère.
2. Écrire la fonction **bien_parenthesee(code : str) → bool** qui renvoie `True` si la ligne de code `code` est correctement parenthésée.
3. Écrire la version récursive **bien_parenthesee_rec(code : str, i : int, p : Pile) → bool**.

Exercice 5 : L'écriture polonaise inverse des expressions arithmétiques place l'opérateur après ses opérandes. Cette notation ne nécessite aucune parenthèse ni aucune règle de priorité. Ainsi l'expression polonaise inverse

$$1\ 2\ 3\ \times\ +\ 4\ \times$$

désigne l'expression traditionnelle

$$(1 + 2 \times 3) \times 4$$

En utilisant une pile pour stocker les résultats intermédiaires, il est facile de calculer l'expression :

- Si on a un nombre on le place sur la pile.
- Si on a un opérateur on récupère les deux nombres du sommet de la pile, on leur applique l'opérateur et on replace le résultat sur la pile.

Écrire la fonction `polonaise(chaine: str)` → `int` qui calcule l'expression *chaine*. Les éléments de la chaîne seront séparés par un espace. On n'utilisera que l'addition et la multiplication.
La méthode `split` permet de découper une chaîne de caractère (voir documentation).

Christophe Viroulaud