Vérification syntaxique de parenthèses ou de balises

> D'après 2022, Métropole, J1, Ex. 1
> **Mise enforme** : Franck Chambon

Partie A : Expression correctement parenthésée

On veut déterminer si une expression arithmétique est correctement parenthésée.
À chaque parenthèse fermante ")" correspond une parenthèse précédemment ouverte "(".

"Exemples"

- L'expression arithmétique `"(2 + 3) × (18/(4 + 2))"` est correctement parenthésée.
- L'expression arithmétique `"(2 + 3) × (18/(4 + 2"` est non correctement parenthésée.

Pour simplifier les expressions arithmétiques, on enregistre, dans une structure de données, uniquement les parenthèses dans leur ordre d'apparition. On appelle expression simplifiée cette structure.

Expression arithmétique	Structure de données
`"(2 + 3) × (18/(4 + 2))"`	`()(())`

1. Indiquer si la phrase « les éléments sont maintenant retirés (pour être lus) de cette structure de données dans le même ordre qu'ils y ont été ajoutés lors de l'enregistrement » décrit le comportement d'une file ou d'une pile. Justifier.

Pour vérifier le parenthésage, on peut utiliser une variable `controleur` qui :

- est un nombre entier égal à 0 en début d'analyse de l'expression simplifiée ;
- augmente de 1 si l'on rencontre une parenthèse ouvrante `"("` ;
- diminue de 1 si l'on rencontre une parenthèse fermante `")"` .

"Exemple"

> On considère l'expression simplifiée A : `"()(())"`
>
> Lors de l'analyse de l'expression A, `controleur` (initialement égal à 0) prend successivement pour valeur 1, 0, 1, 2, 1, 0.
>
> Le parenthésage est correct.

2. Écrire, pour chacune des 2 expressions simplifiées B et C suivantes, les valeurs successives prises par la variable `controleur` lors de leur analyse.

- Expression simplifiée B : " ((()()"
- Expression simplifiée C : "(()))("

3. L'expression simplifiée B précédente est mal parenthésée (parenthèses fermantes manquantes) car le `controleur` est différent de zéro en fin d'analyse.
L'expression simplifiée C précédente est également mal parenthésée (parenthèse fermante sans parenthèse ouvrante) car le `controleur` prend une valeur strictement négative pendant l'analyse.

Recopier et compléter uniquement les lignes 13 et 16 du code ci-dessous pour que la fonction `parenthesage_correct` réponde à sa description.

```
def parenthesage_correct(expression):
    """ fonction renvoyant True si l'expression arithmétique
    simplifiée (str) est correctement parenthésée, False sinon.
    Condition: expression ne contient que
      des parenthèses ouvrantes et fermantes
    """
    controleur = 0
    for parenthese in expression:  # pour chaque parenthèse
        if parenthese == '(':
            controleur = controleur + 1
        else:  # parenthese == ')'
            controleur = controleur - 1
            if controleur ... :  # test 1 (à recopier et compléter)
                # parenthèse fermante sans parenthèse ouvrante
                return False
    return controleur ...  # test 2 (à recopier et compléter)
    # test 2 est un booléen renvoyé
    #   True : le parenthésage est correct
    #   False : parenthèse(s) fermante(s) manquante(s)
```

Partie B : Texte correctement balisé

On peut faire l'analogie entre le texte simplifié des fichiers HTML (uniquement constitué de balises ouvrantes `#!html <nom>` et fermantes `#!html </nom>`) et les expressions parenthésées.

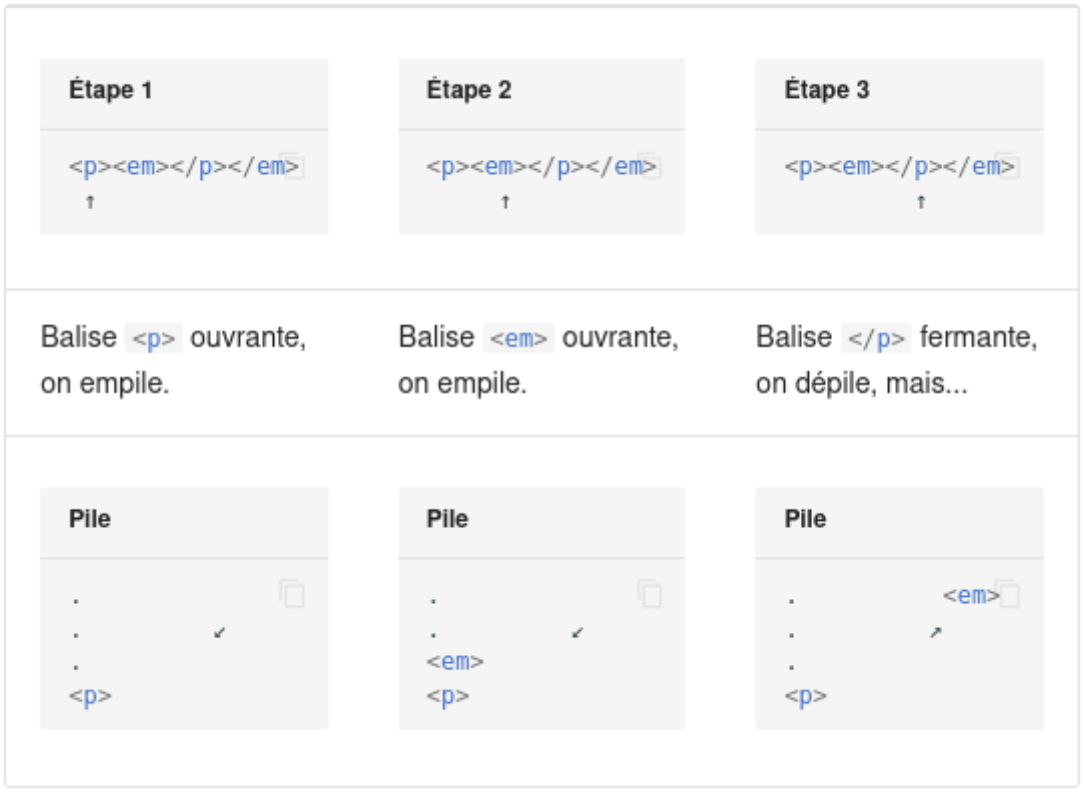
Par exemple, l'expression HTML simplifiée : `#!html "<p><strong><em></em></strong></p>"` est correctement balisée.

On ne tiendra pas compte dans cette partie des balises ne comportant pas de fermeture comme `#!html <br>` ou `#!html <img ...>` .

Afin de vérifier qu'une expression HTML simplifiée est correctement balisée, on peut utiliser une pile (initialement vide) selon l'algorithme suivant :

- On parcourt successivement chaque balise de l'expression :
  - lorsque l'on rencontre une balise ouvrante, on l'empile ;
  - lorsque l'on rencontre une balise fermante :
    - si la pile est vide, alors l'analyse s'arrête : le balisage est incorrect,
    - sinon, on dépile et on vérifie que les deux balises (la balise fermante rencontrée et la balise ouvrante dépilée) correspondent (c'est-à-dire ont le même nom) si ce n'est pas le cas, l'analyse s'arrête (balisage incorrect).

Exemple détaillé



`#!html <em>` et `#!html </p>` ne correspondent pas !
Donc le balisage est incorrect.

4. Cette question traite de l'état de la pile lors du déroulement de l'algorithme.

4.a. Représenter la pile à chaque étape du déroulement de cet algorithme pour l'expression `"<p><em></em></p>"` (balisage correct).

4.b. Indiquer quelle condition simple (sur le contenu de la pile) permet alors de dire que le balisage est correct lorsque toute l'expression HTML simplifiée a été entièrement parcourue, sans que l'analyse ne s'arrête.

5. Une expression HTML correctement balisée contient 12 balises.

Indiquer le nombre d'éléments que pourrait contenir au maximum la pile lors de son analyse.