On dispose de chaînes de caractères contenant uniquement des parenthèses ouvrantes et fermantes.

Un parenthésage est correct si :

- le nombre de parenthèses ouvrantes de la chaîne est égal au nombre de parenthèses fermantes.
- en parcourant la chaîne de gauche à droite, le nombre de parenthèses déjà ouvertes doit être, à tout moment, supérieur ou égal au nombre de parenthèses déjà fermées.

Ainsi, `"((()())(()))"` est un parenthésage correct.
Les parenthésages `"())(()"` et `"(())(()"` sont, eux, incorrects.

On dispose du code de la classe `Pile` suivant :

```
class Pile:
    """Classe définissant une pile"""
    def __init__(self, valeurs=[]):
        self.valeurs = valeurs

    def est_vide(self):
        """Renvoie True si la pile est vide, False sinon"""
        return self.valeurs == []

    def empiler(self, c):
        """Place l'élément c au sommet de la pile"""
        self.valeurs.append(c)

    def depiler(self):
        """
        Supprime l'élément placé au sommet de la pile, à condition qu'elle
        soit non vide
        """
        if self.est_vide() == False:
        self.valeurs.pop()
```

On souhaite programmer une fonction `parenthesage` qui prend en paramètre une chaîne `ch` de parenthèses et renvoie `True` si la chaîne est bien parenthésée et `False` sinon.
Cette fonction utilise une pile et suit le principe suivant : en parcourant la chaîne de gauche à droite, si on trouve une parenthèse ouvrante, on l'empile au sommet de la pile et si on trouve une parenthèse fermante, on dépile (si possible !) la parenthèse ouvrante stockée au sommet de la pile.
La chaîne est alors bien parenthésée si, à la fin du parcours, la pile est vide.
Elle est, par contre, mal parenthésée :
- si dans le parcours, on trouve une parenthèse fermante, alors que la pile est vide ;
- ou si, à la fin du parcours, la pile n'est pas vide.

```
def parenthesage (ch):
    """Renvoie True si la chaîne ch est bien parenthésée et False sinon"""
    p = Pile()
    for c in ch:
        if c == ...:
            p.empiler(c)
        elif c == ...:
            if p.est_vide():
                return ...
        else:
            ...
    return p.est_vide()

assert parenthesage("((()())(()))") == True
assert parenthesage("())(()") == False
assert parenthesage("(())(()") == False
```

Compléter le code de la fonction `parenthesage`.