**Corrigé n° 9.4 de l'exercice 9.4**

1. **En opérant à la main** (si vous ne le faites pas, vous ne comprendrez pas ce qui suit), on constate qu'en cas de mauvais parenthésage, la pile est soit vide au moment de dépiler, soit non vide à la fin du texte déroulé. S'il y a plusieurs types de parenthèses ou crochets et accolades, il nous faudra en plus tester qu'au moment où l'on rencontre une fermante, c'est bien l'ouvrante du même type qui figure au sommet de la pile.
2. Cela nous incite à interrompre le processus lorsque la pile est prématurément vide et à tester l'état de la pile en fin de texte :

```
def ouvrante(p):
      return  p=='(' or p=='[' or p=='{'

def fermante(p):
      return  p==')' or p==']' or p=='}'

def renverse(p):
      if p ==')':
              return '('
      if p ==']':
              return '['
      if p =='}':
              return '{'
```

```
import PilesMin as pm

def  verifier_parentheses(texte):
  p = pm.creer_pile()
  for i in range(0,  len(texte) ):
    if ouvrante(texte[i]):
       pm.empiler(p,texte[i])
    elif fermante(texte[i]):
       if pm.est_vide(p) or
          renverse(texte[i]) != pm.sommet(p):     (*)
         return False, ' Mauvais ... : ' + texte
       else:
         pm.depiler(p)
  if pm.est_vide(p):
      return  True, 'Parenthésage correct '
  else:
      return   False, ' Mauvais ... : ' + texte


>>> verifier_parentheses('{5+7}*12')
(True, 'Parenthésage correct ')
>>> verifier_parentheses(' {un (deux (trois}}}')
(False, ' Mauvais... dans:  {un (deux (trois}}}')
>>> verifier_parentheses(' {un (deux [trois])}')
(True, 'Parenthésage correct ')
>>> verifier_parentheses(' {un (deux [trois)]}')
(False, ' Mauvais... dans:  {un (deux [trois)]}')
```

3. ★ Si nous voulons prouver la correction de la fonction, **commençons par définir ce qu'est un bon parenthésage.** Passons donc au niveau supérieur et explicitons. Nous dirons qu'une chaîne de caractères est bien parenthésée lorsque :
   – soit, elle ne contient pas de parenthèse ;
   – soit, elle contient au moins deux parenthèses, la première et la dernière sont respectivement ouvrante et fermante, donc de sens opposés, et de même type, et la sous-chaîne contenue entre ces parenthèses est elle-même bien parenthésée.

   On observe tout d'abord qu'un bon parenthésage demande un nombre pair de parenthèses et autant d'ouvrantes que de fermantes. Or le programme retourne False si le nombre de parenthèses est impair (en effet, soit la boucle est interrompue avec une instruction return False, soit il y a autant d'appels à empiler

qu'à dépiler et comme le nombre de parenthèses est impair, la pile ne sera pas vide et la fonction renvoie bien False).

Il reste à vérifier que lorsqu'il y a un nombre pair de parenthèse le résultat est conforme à la définition.

Montrons alors par récurrence le résultat

$\mathcal{P}(n) = \{$la fonction renvoie un résultat correct lorsqu'il y a 2n parenthèses$\}$.

- Si $n = 0$, rien n'est fait dans la boucle, la pile reste vide, la fonction renvoie True. Correct.
- Supposons que le résultat soit correct pour une chaine de $2n$ parenthèses.
  - Sur une chaîne ayant $2n + 2$ parenthèses, et correctement parenthésée il y aura par définition une première parenthèse ouvrante en position $i_1$ qui sera empilée, ensuite le parcours de la chaîne entre $i_1 + 1$ et $i_{2n+2} - 1$ est celui d'une **chaîne bien parenthésée** et les parenthèses empilées sont dépilées sans accroc par hypothèse de récurrence (algorithme inchangé). Arrivé en $i_{2n+2}$ rien ne se passe plus alors jusqu'à la fin de la chaîne. La pile est vide, le résultat, True est celui que l'on attend.
  - Sur une chaîne ayant $2n + 2$ parenthèses, et mal parenthésée, soit au cours de la lecture entre $i_1 + 1$ et $i_{2n+2} - 1$ le programme s'arrête (sur un return) et alors par hypothèse de récurrence c'est que la sous chaîne est incorrecte, soit on arrive à la lecture du caractère $i_{2n+2}$ parce que la sous-chaîne est correcte, mais le résultat est False à ce moment là puisque les deux parenthèses extrêmes sont celles qui mettent le parenthésage en défaut (mauvais sens ou de types différents : le programme le détecte...).

**Remarque** : lorsque le nombre de parenthèses est pair, si le parenthèsage est malgré cela incorrect, la boucle est toujours interrompue avant le dernier if-else...

**Observation** : c'est plus que ce que l'on peut vous demander (et que ce que l'on éprouve le besoin de faire au quotidien) !