TP 6 – Piles et fonctions récursives

Présentation du problème

Une pile en informatique est un format de donnée similaire à une liste, mais, comme une pile d'assiettes, ne peut être manipulée que selon les cinq fonctions dont un listing est donnée ci-dessous :

```
1  ######################################################
2
3  # Fonctions propres aux piles construites Ã  partir des fonctions
       sur les listes
4
5  def creer_pile():
6      return[]
7
8  # Teste si une pile est vide (1 boulÃ©en oui, 0 boulÃ©en non)
9  def empty(P):
10     return P==[]
11
12 # ajoute l'Ã©lÃ©ment x Ã  la pile P
13 def push(P,x):
14     P.append(x) # on rappelle que liste.append(x) ajoute x Ã  la
       fin de liste
15
16 # supprime le dernier Ã©lÃ©ment de la pile
17 def pop(P):
18     if empty(P): # utilise la fonction empty qui vient d'Ãªtre
       crÃ©Ã©e
19         print("erreur : pile vide")
20         return ("erreur")
21     else:
22         del P[-1] # supprime le dernier Ã©lÃ©ment de P (selon l'
       idÃ©e d'une liste cylique)
23         return(P)
24
25 # indique quel est l'Ã©lÃ©ment au sommet de la pile P
26 def sommet(P):
27     if empty(P):
28         print("erreur : pile vide")
29         return ("erreur")
30     else:
31         return P[-1]
32
33 ######################################################
```

1 Quelques exercices classiques sur les piles

Exercice 1 — Un peu de verlan

1. Écrire un programme prenant en entrée une chaîne de caractères et affichant ces caractères dans l'ordre inverse. On utilisera une pile.
2. Écrire un programme qui teste si une chaîne de caractères est un palindrome, c'est à dire un mot ou une expression qui est identique qu'on la lise de gauche à droite ou de droite à gauche.

Exercice 2 — Hauteur de pile

1. Écrire une fonction `Hauteur(P)` qui calcule le nombre d'éléments dans une pile `P`.
   Lorsque cette fonction se termine la pile `P` devra avoir retrouvé son état initial.
2. Écrire une fonction `Copie(P1)` qui prend en entrée une pile `P1` et retourne une copie `P2` de la pile `P1` . Lorsque cette fonction se termine, la pile `P1` doit avoir retrouvé son état initial.

Exercice 3 — Bons parenthésages

1. Lire les expressions littérales suivantes, et à l'aide d'une pile, tester manuellement si elles sont ou non bien parenthésées (*on donnera les états successifs de la pile à chaque fois qu'un parenthèsage est repéré*).
   a. $[a + \{b/(c - d) + e/(f + g)\} - h]$
   b. $[a * \{b + c * (d - e)/f + \{g - h\}]$
   c. $[a * \{b + c * (d + e) - f\} + g] - h]$

2. Écrire une fonction qui prend en argument une chaîne de caractères représentant une expression mathématique parenthésée et qui teste si cette expression mathématique est bien parenthésée.

2 Écriture en polonaise inverse

La notation polonaise inverse est une écriture des opérations utilisée par certaines calculatrices (souvent anciennes). On l'appelle aussi notation postfixée car elle consiste à écrire les opérateurs après leurs opérandes. Bien que peu usuelle pour les occidentaux, cette notation permet d'écrire des calculs algébriques sans avoir besoin de recourir à des parenthésage. De plus, avec de l'habitude la notation polonaise inverse permet de calculer plus vite que la notation usuelle.

Exemples

- l'expression $3 + 7$ s'écrit $3\ 7\ +$
- L'expression $(3 + 7) \times 2$ s'écrit $3\ 7\ +\ 2\ \times$ mais aussi $2\ 3\ 7\ +\ \times$

Pour calculer en notation polonaise inverse on une pile des opérandes. Quand on évalue une expression postfixée, on empile les nombres dès qu'on les voit apparaître. Par contre quand on lit un opérateur :

- on désempile les deux nombres au sommet de la pile des opérandes ;

- on effectue l'opération sur ces deux nombres ;
- on empile le résultat du calcul

Exercice 4 — Implémenter une fonction `Evaluation (L)` qui prend en entrée une liste contenant soit des nombres soit des opérateurs (décrivant une expression algébrique en notation polonaise inverse), et qui retourne la valeur de l'évaluation de cette expression.
Pour simplifier, on supposera que seuls deux opérateurs apparaissent : $+$ et $*$.

Pour écrire une expression usuelle (dite ***Infix***) en notation postfixée, on utilise une pile des parenthères et opérateurs :

- On commence par mettre une parenthèse `(` dans la pile
- lorsqu'on lit un nombre, on le met tout de suite dans l'expression postfixée
- Lorsqu'on lit un opérateur `*` ou une parenthèse `(` on le met dans la pile
- Losqu'on lit un opérateur + et que le sommet de la pile n'est pas `*` on empile l'opérateur `+`
- Losqu'on lit un opérateur + et que le sommet de la pile est `*` on désempile la pile d'opérateurs et on met * dans l'expression postfixée

  On continue cette action de désempilage jusqu'à ce que le sommet de la pile ne soit plus `*`, quand c'est le cas, on empile `+`
- Lorsqu'on lit une parenthèse `)` on désempile la pile (et on met les caractères désempilés dans l'expression posfixée) jusqu'à ce que le l'on ait désempilé une parenthèse `(`.

  Attention : on **ne met pas** de parenthèse dans l'expression postfixée !

Exemple
On convertit l'expression

$$(a + (b + c * d + b * a) * b * c)$$

Les opérations effectuées sont en dernière page.

Exercice 5 — écrire une fonction `Postfix(L)` qui transforme une liste contenant une expression écrite en notation Infix et retourne une liste contenant la conversion de cette expression en notation postfix.

3 Tours de Hanoï

Le jeu des tours de Hanoï est constitué de trois tiges (représentées par des piles). Sur l'une des tiges sont empilés n disques de tailles différentes, de sorte que chaque disque soit posé sur un disque plus grand que lui.
Le but du jeu est de déplacer tous les disques de la tige de départ vers une autre tige, en déplaçant un seul disque à la fois, et en ne posant jamais un disque sur un plus petit que lui.
écrire une fonction `Hanoi(T1, T2, T3,n)` qui indique la suite des mouvements à effectuer pour déplacer n disques, initialement posés sur la tige $T1$ portant le numéro 1, vers la tige $T3$ portant le numéro 3 (en utilisant la tige $T2$ portant le numéro 2).

Par exemple, on devra avoir

```
>>> Hanoi("A","B","C",3)
Deplacer un disque depuis  A  vers  C
Deplacer un disque depuis  A  vers  B
Deplacer un disque depuis  C  vers  B
Deplacer un disque depuis  A  vers  C
Deplacer un disque depuis  B  vers  A
Deplacer un disque depuis  B  vers  C
Deplacer un disque depuis  A  vers  C
```

Opérations effectuées (*Exercice 4*)

Caractère lu	**état de la pile**	**Expression post–fixée**
(	(	
a	(	a
$+$	$(+$	a
(	$(+($	a
b	$(+($	ab
$+$	$(+(+$	ab
c	$(+(+$	abc
$*$	$(+(+*$	abc
d	$(+(+*$	$abcd$
$+$	$(+(++$	$abcd*$
b	$(+(++$	$abcd*b$
$*$	$(+(++*$	$abcd*b$
a	$(+(++*$	$abcd*ba$
)	$(+$	$abcd*ba*++$
$*$	$(+*$	$abcd*ba*++$
b	$(+*$	$abcd*ba*++b$
$*$	$(+**$	$abcd*ba*++b$
c	$(+**$	$abcd*ba*++bc$
)		$abcd*ba*++bc**+$

$\sim$