# TD – Piles et files

## Introduction et objectifs

Dans ce TD consacré principalement aux piles (stacks) et, dans une moindre mesure, aux files (queues), on doit fondamentalement tenir compte des contraintes imposées par les structures des piles et des files. En particulier, dans les exercices 1 à 4 ci-dessous, on utilisera exclusivement les fonctions de création et manipulation de piles vues en cours et mentionnées dans le « cahier des charges ».

## Piles

### Exercice N°1 – Copie d'une pile

Ecrire une fonction `stack_copy` (ou une méthode `copy`) recevant une pile `s` comme argument et renvoyant une copie `s2` de `s`. Attention, la pile `s` doit (bien sûr…) être conservée !

Evaluer le coût en mémoire et le nombre d'opérations de la fonction.

### Exercice N°2 – Inversion d'une pile

Ecrire une fonction `stack_reverse` (ou une méthode `reverse`) recevant une pile (`s`) comme argument et renvoyant une copie inversée `rs` de `s`. Attention, la pile `s` doit être conservée !

Evaluer le coût en mémoire et le nombre d'opérations de la fonction (on envisagera ici deux cas suivant que l'on conserve ou non la pile initiale `s`).

### Exercice N°3 – Permutations circulaires (acte 1)

Ecrire une fonction `stack_circperm` (ou une méthode `circperm`) qui reçoit en argument une pile `s` et un entier `n` et effectue sur la pile `n` permutations circulaires successives.

Dans cet exercice, c'est la pile `s` elle-même qui sera modifiée.

Exemple avec `n=2` :

| | | |
|---|---|---|
| 7 | | 98 |
| 11 | | 2 |
| 98 | donnera | 103 |
| 2 | | 7 |
| 103 | | 11 |

Evaluer le coût en mémoire et le nombre d'opérations de la fonction.

## Exercice N°4 – Permutations circulaires (acte 2)

Ecrire une fonction `stack_circperm2` (ou une méthode `circperm2`) qui reçoit en argument une pile `s` et un entier `k` et effectue une permutation circulaire sur les `k` premiers éléments de la pile.

Dans cet exercice, c'est la pile `s` elle-même qui sera modifiée.

Exemple avec `k=4` :

| | | |
|---|---|---|
| 7 | | 11 |
| 11 | | 98 |
| 98 | donnera | 2 |
| 2 | | 7 |
| 103 | | 103 |
| 5 | | 5 |

Evaluer le coût en mémoire et le nombre d'opérations de la fonction.

## Exercice N°5 – Expression correctement parenthésée

Dans un logiciel de calcul formel ou, plus généralement dans un éditeur de texte (par exemple utilisé pour écrire des programmes), il y a une gestion dynamique (i.e. « au vol ») du parenthésage : si une parenthèse fermante de trop est ajoutée ou si elle n'est pas de même nature (une accolade au lieu d'un crochet par exemple) que la parenthèse ouvrante associée alors un message d'erreur (visuel, sonore) est envoyé.

Par exemple, les deux expressions suivantes sont erronées :

$$A = \left(4 + i\sqrt{3}\right)\Big)^{17}$$

(deux parenthèses fermantes pour une ouvrante.)

$$B = \sin\left\{\left[1+\frac{3}{n}\right)^{n}\right\}$$

(la première parenthèse fermante n'est pas de même nature
que la deuxième « parenthèse » ouvrante.)

Ecrire une fonction qui reçoit comme argument une chaîne de caractères, en analyse la correction du parenthésage et renvoie à l'utilisateur un message adapté.

L'analyse de la chaîne de caractères se fera caractère après caractère. On gèrera une pile contenant les parenthèses ouvrantes et on comparera chaque parenthèse fermante au sommet (éventuel) de la pile.

## Files

### Exercice N°6

Reprendre les exercices 1 et 3 de la partie « Piles » mais en traitant cette fois le cas d'une file.

## Suggestions de compléments

1. Les fonctions `stack_circperm` et `stack_circperm2` peuvent être réunies en une seule. La nouvelle fonction recevra trois arguments : le nom de la pile, le nombre d'éléments sur lesquels on appliquera des permutations circulaires et, enfin, le nombre de permutations circulaires à appliquer à ces éléments.
2. En Python, les concepts de pile et de file sont généralisés via les objets `deque` (double ended queue) du module `collections`. Ce sont en quelque sorte des files modifiables à gauche et à droite.

   Les méthodes utilisables et de nombreux exemples se trouvent au paragraphe :

   8.3.2 de la documentation Python (versions 2.x) :
   http://docs.python.org/2/library/collections.html?highlight=deque#deque-objects

   8.3.3 de la documentation Python (versions 3.x) :
   http://docs.python.org/3/library/collections.html?highlight=deque#deque-objects