TD 2

Exercices d'applications

Savoirs et compétences :
- ❒ *Alg – C16 : Piles - Algorithmes de manipulation : fonctions «push» et «pop».*

Exercice 1 – Construction d'une pile

On souhaite réaliser les fonctions de base permettant la gestion d'une pile. On souhaite que celle-ci soit implémentée sous la forme d'une liste. La taille de la pile doit être limitée à `n` données.

Ainsi, une pile de taille 5 avec 4 éléments sera de la forme suivante : `[1,2,3,4,None]`.

Question 1 *Implémenter la fonction* `creer_pile(n)` *permettant de créer une pile de taille n. Les spécifications seront les suivantes :*

▪ Python
```
def creer_pile(n):
    """
    Créer une pile de taille n.
    Entrée :
     * n(int) : taille souhaitée de la pile
    Sortie :
     * pile(list) : pile de taille n.
     sinon
    """
```

Question 2 *Implémenter la fonction* `est_vide` *permettant de vérifier qu'une pile est vide. Les spécifications de la fonction seront les suivantes :*

▪ Python
```
def est_vide(pile):
    """
    Vérifie si la pile est vide.
    Entrée :
     * pile(list)
    Sortie :
     * retourne True si la pile est vide, False
     sinon
    """
```

Question 3 *Implémenter la fonction* `est_pleine` *permettant de vérifier qu'une pile est pleine. Les spécifications de la fonction seront les suivantes :*

▪ Python
```
def est_pleine(pile,nb):
    """
    Vérifie si la pile est pleine.
    Entrée :
     * pile(list)
     * nb(int) : nombre d'éléments maximum dans une pile
    Sortie :
     * retourne True si la pile est pleine, False
     sinon
    """
```

Question 4 *Créer la fonction* `empiler` *permettant d'ajouter un élément au sommet de la pile. Les spécifications de la fonction devront être indiquées.*

Question 5 *Créer la fonction* `depiler` *permettant de supprimer l'élément au sommet de la pile et de le renvoyer. Les spécifications de la fonction devront être indiquées.*

Question 6 *Créer la fonction* `taille_pile` *permettant de donner la longueur de la pile.*

Exercice 2

Écrire une fonction concaténant deux piles.
Cette fonction prendra en argument deux piles et placera la première « sur » la deuxième. Elle ne renverra rien, mais modifiera les deux piles : à la fin, la première pile sera vide et la seconde aura été augmentée de la première. Pour cela, on supposera que la seconde pile est assez grande pour recevoir toute la première.

On proposera une version itérative et récursive.

Exercice 3

Écrire une fonction renvoyant la plus grande somme possible de trois éléments consécutifs d'une pile d'entiers naturels. Si la pile a deux éléments ou moins, on renverra 0. On prendra garde de recopier la pile passée en argument afin de ne pas la modifier.

Exercice 4

On se donne deux piles P_1 et P_2, et on se propose de fusionner ces deux piles en une seule pile P de manière à respecter les règles suivantes :

1. Si $P_1 = [a_1, a_2, \cdots, a_{n-1}, a_n]$ et $P_2 = [b_1, b_2, \cdots, b_{p-1}, b_p]$, alors $P = [\cdots, b_{p-1}, a_{n-1}, b_p, a_n]$.
2. Si l'une des pilees est vide, alors on n'a plus recours qu'à l'autre, jusqu'à ce que celle-ci aussi soit vide.

Exemples : $P_1 = (a_1, a_2, a_3)$ et $P_2 = (b_1, b_2, b_3, b_4, b_5)$, alors $P = [b_1, b_2, a_1, b_3, a_2, b_4, a_3, b_5]$.

Écrire une fonction python réalisant cela. On prendra garde de recopier les piles passées en argument afin de ne pas la modifier.
Afin d'être sûr que la nouvelle pile est assez grande pour contenir les deux premières, on pourra commencer par écrire une fonction recopiant une pile dans une autre pile de taille donnée.

Exercice 5 – Notation polonaise inversée

La notation polonaise inversée permet de réaliser des opérations arithmétiques sans utiliser de parenthèses. On parle aussi d'expressions **postfixées**. Dans cette notation, l'opérateur (+, -, *, /) suit toujours l'opérande (nombres). Ici, l'opération suit le deuxième opérande.

On doit évaluer une expression composée d'opérateurs et d'opérandes. L'expression est lue de gauche à droite en suivant les règles suivantes :

- si on lit une opérande, on l'empile ;
- si on lit un opérateur, on dépile deux opérandes, on réalise le calcul et on empile le résultat.

▪ **Exemple** Pour calculer $7 \cdot 2$, on évalue l'expression $7\,2\,*$.

Pour calculer $((1+2)\cdot 4)+3$, on évalue l'expression $1\,2\,+\,4\,*\,3\,+$. ▪

R Le principal avantage de cette méthode est la suppression totale des parenthèses. En contrepartie, elle nécessite une petite gymnastique intellectuelle pour les néophytes que nous sommes.

Cette notation est utilisée dans certaines calculatrices HP et dans certains utilitaires (programme calc d'Emacs, description des bibliographies dans LaTeX *etc.*).

Dans cet exercice, la pile sera implémentée sous forme d'une liste d'opérande et d'opérateurs.

On utilisera les fonctions définies à l'exercice précédent.

Question 1 *Créer la fonction* `est_nombre` *permettant de savoir si une variable est un nombre ou non.*

Question 2 *Créer la fonction* `est_operation` *permettant de savoir si une variable est une opération ou non.*

Question 3 *L'expression postfixée se lisant de gauche à droite et les outils permettant de traiter la pile traitant cette dernière de droite à gauche, réaliser la fonction* `inversion` *permettant d'inverser les éléments de la pile.*

Question 4 *Créer la fonction* `evaluer` *permettant d'évaluer une expression postfixée. Les spécifications sont les suivantes :*

▪ **Python**

```
def evaluer(exp):
    """
    Évaluer le résultat d'une opération post-fixée.
    Entrée :
     * ex(lst) : liste d'opérateurs et d'opérandes
    Sortie :
     * res(flt) : résultat du calcul de l'expression.
    """
```

Question 5 *Transcrire l'expression suivante en NPI puis l'évaluer :* $5+((1+2)\times 4)-3$.