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):
    """
    Crêer 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 – 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 :*

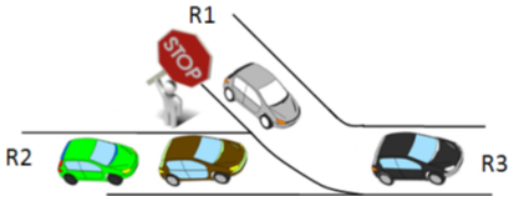
▪ Python
```
def evaluer(exp):
    """
    Évaluer le résultat d'une opération post-fixée.
    Entrée :
     * ex(lst) : liste d'opérateurs et d'opérandes
    Sortie :
     * res(flt) : résultat du calcul de l'expression.
    """
```

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

Exercice 3 – Croisement routier

D'après P. Beynet.

Pour simuler un croisement routier, à sens unique, on utilise 3 files `f1`, `f2` et `f3` représentant respectivement les voitures arrivant sur des routes `f1` et `R2`, et les voitures partant sur la route `R3`. La route `R2` a un STOP. Les voitures de la file `f2` ne peuvent avancer que s'il n'y a aucune voiture sur la route `R1`, donc dans la file `f1`. On souhaite écrire un algorithme qui simule le départ des voitures sur la route `R3`, modélisée par la file `f3`.



On utilise trois files `f1`, `f2` et `f3` de valeurs booléennes :

- « 1 » symbolise la présence d'une voiture ;
- « 0 » l'absence de véhicule.

Question 1 *Réaliser l'algorithme de croisement routier permettant de simuler la composition de la file* `f3` *en fonction de la composition des files* `f1` *et* `f2`*. Les files seront modélisées par des listes de booléens.*