Informatique en CPGE (2018-2019)
TD 1 : notion de pile

Exercice 1

Nous allons travailler sur une pile à capacité finie.

1. Ecrire le code des fonctions **creer_pile**, **empiler**, **depiler** vues en cours.
2. Créer une pile de capacité 4.
3. Compléter le code en utilisant la fonction **empiler** afin d'obtenir la pile contenant dans l'ordre les caractères 'A', 'B', 'C', 'D'. Vérifier que la liste représentant la pile est [4,'A','B','C','D'].
4. Ecrire une fonction **vider_pile(p)** qui prend en argument une pile p et renvoie la pile vidée.
5. Quel est le résultat après chaque opération dans la suite d'instructions empiler(p,4), empiler(p,1), empiler(p,3), depiler(p), empiler(p,8), et depiler(p) sur une pile initialement vide ? Prévoir le résultat et le vérifier avec le programme.

Exercice 2 : notation polonaise inverse

L'objectif est d'écrire un programme pour évaluer des expressions écrites en notation polonaise inverse. Les expressions en NPI seront représentées par des tableaux contenant des entiers et des caractères. Par exemple, « 2 3 + 7 × » sera représenté par le tableau [2, 3, '+', 7, '*'] .

Principes du programme : une pile vide est créée ; le tableau est parcouru de gauche à droite et chaque nombre rencontré est empilé. Si l'élément rencontré est un opérateur, les deux opérandes sont désempilés et le résultat du calcul est empilé.

1. Nous utilisons une pile à capacité finie ; commencer par importer le fchier dans lequel sont écrites les fonctions **creer_pile(c)**, **empiler(p,x)**, **depiler(p)**, **pile_vide(p)** vues en cours.
2. Les tableaux sont représentés par des objets de type **list**. Ecrire une fonction **calc_npi(exp)** qui prend en argument une liste notée **exp**. Nous ne considérons ici que les opérateurs "+" et "*". Le corps de cette fonction se compose ainsi :
   (a) création d'une pile p dont la capacité maximale est la longueur de l'expression arithmétique ;
   (b) parcourt avec une boucle **for** des éléments de **exp**, de gauche à droite ;
   (c) si l'élément courant est un opérateur (caractère '+' ou '*'), on dépile les deux opérandes x et y de p et on empile x + y ou x * y , selon la valeur de c :
   Sinon, c est un nombre et on l'empile dans p :
   (d) quand on sort de la boucle for , il ne reste plus qu'à dépiler la valeur finale res de l'expression
   (e) vérifier que la pile p est bien vide :
3. Vérifier le fonctionnement sur quelques exemples :
   calc_npi([3, 4, '+', 5, '*']) (réponse : 35) ;
   calc_npi([3, 4, '*', 5, '+']) (réponse : 17) ;
   calc_npi([3, 4, '+', 5, '+']) (réponse : 12).
4. Ajouter à la fonction calc_npi la soustraction et la division.
5. Ajouter à la fonction calc_npi le calcul de la valeur absolue, du carré et de l'opposé d'un entier.
6. Modifier le programme afin que l'expression à calculer soit entrée au clavier par l'utilisateur soit en une seule fois (ce sera donc une chaîne de caractères, par exemple avec des espaces "12 23 +"), soit élément par élément à la demande (chaque élément sera une chaîne de caractère et le critère d'arrêt sera l'entrée du signe '=').

Exercice 3 : tour de Hanoï

On dispose d'une pile à capacité finie p1=[4,4,3,2,1], où p1[0]=4 est le nombre d'éléments de la pile.

1. Utiliser la fonction **creer_pile** pour créer deux piles vides p2 et p3 de capacité 4.
2. L'objectif est d'obtenir une pile p3 identique à la pile p1 initiale, soit p3=[4,4,3,2,1] en un minimum de "coups".

   Un "coup" est le déplacement d'un nombre, sommet d'une pile, sur une autre pile. Pour ce faire, on utilisera les fonctions **depiler** et **empiler**.

   Règle : on empile un nombre sur une pile uniquement s'il est plus petit que le sommet de la pile.

   Ecrire le programme permettant d'obtenir le résultat souhaité.

   On vérifiera que le minimum de coups est $15 = 2^4 - 1$.
3. Ecrire un programme qui résout ce problème "au hasard".

   Principe : tant que les piles p1 et p2 ne sont pas vides, on choisit au hasard une des trois piles ; si cette pile n'est pas vide, on dépile son sommet et on choisit au hasard une des deux autres piles sur laquelle on empile l'élément seulement si c'est possible.

   Vérifier que le nombre de coups moyens pour résoudre le problème avec p1=[4,4,3,2,1] et des piles de capacité 4 est environ 530.

   Tester le programme avec p1=[5,5,4,3,2,1] et des piles de capacité 5, puis p1=[6,6,5,4,3,2,1] et des piles de capacité 6, puis ...