TD: Structures linéaires-piles et files

Piles

Exercice N°1 – Copie d'une pile

Ecrire une fonction **`copyPile(s)`** recevant une pile (**`s`**) comme argument et renvoyant une copie **`s2`** de **`s`**. Attention, la pile **`s`** doit être conservée !

Exercice N°2 – Inversion d'une pile

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

Exercice N°3 – Permutations circulaires

Ecrire une fonction **`permutCerc(s,n)`** 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
11
98
2
103

donnera

98
2
103
7
11

Une première remarque en guise de préambule : on peut supposer que l'utilisateur (ou le programme appelant) fournisse bien pour `n` un entier naturel. Mais il est possible que cet entier soit plus grand que la longueur de la pile `s`. Ainsi, le nombre effectif de permutations circulaires à mettre en œuvre est en fait égal au reste `r` de la division euclidienne de `n` par **`len(s)`,** soit, en Python : **`r=n%len(s)`**. Il est clair que si ce reste est nul, il convient de ne rien faire !

Illustrons le principe général de l'algorithme à partir de l'exemple fourni dans l'énoncé.

On commence par construire une pile `s2` contenant les `r` premiers éléments de la pile `s` qui est donc successivement dépilée.	On se retrouve ainsi avec les deux piles : **s** : 98, 2, 103 (sommet : 98) **s2** : 7, 11 (sommet : 7)
On inverse alors la pile **`s`** (on note **`rs`** la pile inversée). On pourrait bien sûr utiliser la fonction de l'exercice 2 mais conserver la pile **`s`** ne nous est à priori ici d'aucune utilité.	La pile **`rs`** est donc obtenue directement en dépilant la pile **`s`**. On obtient : **rs** : 103, 2, 98 (sommet : 103) **s2** : 7, 11 (sommet : 7) La pile demandée, que nous notons `cps`, est alors directement obtenue en dépilant successivement **`s2`** puis **`rs`**.

Files

On se donne trois files F1, F2 etF3. La file F1contient des nombres entiers positifs. Les files F2 et F3 sont initialement vides. En n'utilisant que ces trois files (et aucune autre structure de données linéaire):

Q1 : Écrire une fonction recevant la file F1 et retourne la file F2 contenant tous les entiers de F1qui sont des multiples de 5. Faire en sorte que la file F1 soit vide après l'exécution de la fonction.

Q2 : Écrire une fonction qui reçoit la file F1 et retourne une copie F2 de F1. Le contenu de F1 après exécution de la fonction doit être identique à celui avant exécution.

Q3 : Écrire une fonction qui reçoit la file F1 et retourne la file F2 contenant les entiers de F1 qui sont des multiples de 3. Le contenu de F1 après exécution de la fonction doit être identique à celui avant exécution.

Q4 : Écrire une fonction qui reçoit la file F1 et retourne les files F2 et F3, F2 contiendra les entiers pairs de F1 et F3 les entiers impairs de F1. Après l'exécution de la fonction, F1 est vide.

Q5 : Écrire une fonction qui place les entiers de F1 dans F2 en plaçant les entiers impairs derrière les entiers pairs. Après l'exécution de la fonction, F1 est vide et les entiers de F2 sont dans l'ordre initial.

Exemple: si au début F1contient 1, 2, 3, 4 et 5, alors F2 contient 2, 4, 1, 3 et 5 après l'exécution de la fonction.

On se donne maintenant une seule file F non vide et qui contient des nombres entiers positifs. En n'utilisant que cette file (et aucune autre structure de données linéaire):

Q6 : Écrire une fonction qui effectue une permutation circulaire d'ordre1 des entiers de la file F; le 1er élément se retrouve en dernière position, le 2^{ième} élément devient le 1er, etc.

Exemple: si au départ F contient les éléments 1, 2 et 3, après l'exécution F contient 2, 3 et 1.

Q7 : Écrire une fonction qui renvoie `True` si la file contient au moins un entier pair et `False` dans le cas contraire; après l'exécution, la file contient exactement les mêmes éléments qu'au début et dans le même ordre.