# TD 4 : Piles

On rappelle les fonctions agissant sur les objets de type « pile » (**aucune** autre fonction ne devra être utilisée dans ce TD) :

| Désignation | Fonction | Type objet retourné |
|---|---|---|
| `CreerPile()` | crée une pile vide | `pile` |
| `EstVide(p)` | retourne l'état vide ou non vide de la pile `p` | booléen |
| `TaillePile(p)` | retourne la taille de la pile `p` | entier |
| `Empiler(p,elt)` | empile `elt` au sommet de la pile `p` | rien : pile modifiée sur place |
| `Depiler(p)` | dépile le sommet de la pile | rien : pile modifiée sur place |
| `Sommet(p)` | retourne le sommet de la pile | type des éléments de la pile |

## Manipulations de piles

1. Écrire une fonction `copie(p)` recevant une pile `p` comme argument et renvoyant une copie de la pile. La pile `p` doit être conservée. Évaluer le coût en mémoire et le nombre d'opérations de la fonction.
2. Écrire une fonction `inversion(p)` recevant une pile `p` comme argument et renvoyant la pile inversée. La pile `p` doit être conservée. Évaluer le coût en mémoire et le nombre d'opérations de la fonction.
3. Écrire une fonction `permutation(p,n)` qui reçoit en argument une pile `p` et un entier $n$ et effectue sur la pile $n$ permutations circulaires successives. C'est ici la pile `p` elle-même qui sera modifiée.
   Exemple avec $n = 2$ : `[0,12,1,-4,5]` donnera `[-4,5,0,12,1]`
   Évaluer le coût en mémoire et le nombre d'opérations de la fonction.
4. Écrire une fonction `recuperer(p,elt)` qui reçoit en argument une pile `p` et un élément `elt` du type de ceux de la pile `p` et qui renvoie la position de l'élément `elt` dans la pile (position comptée à partir du sommet) ; la pile `p` doit être modifiée en enlevant `elt` ; la fonction renverra `None` si `elt` n'est pas dans la pile `p`. Si `p = [1,2,3,4,3,5]` l'exécution de `recuperer(p,3)` doit renvoyer `1`, la pile `p` doit devenir `[1,2,3,4,5]`.
   Évaluer le coût en mémoire et le nombre d'opérations de la fonction.

## Le jeu du « 4 à la suite »

Le candidat à un jeu TV doit répondre à une suite de questions. Son score est le nombre maximal de bonnes réponses données à la suite. Écrire une fonction `jeu(exp)` qui donne le score du candidat à partir d'une expression de longueur quelconque du type `'VFFVVF'` qui indique dans l'ordre chronologique la vérité des réponses du candidat aux questions (`V` pour vrai et `F` pour faux). La fonction utilise une pile et empile les bonnes réponses mais dépile toute la pile à la première mauvaise réponse. De plus, elle arrête d'examiner les réponses si le candidat a donné 4 bonnes réponses à la suite.

## Trier avec des piles

On considère une pile `p` contenant des entiers que l'on veut trier. On va procéder de la façon suivante : on utilisera deux piles $\mathtt{p}_1$ et $\mathtt{p}_2$ qui doivent vérifier les propriétés suivantes :

- les éléments de $\mathtt{p}_1$ sont empilés en ordre décroissant.
- les éléments de $\mathtt{p}_2$ sont empilés en ordre croissant.
- le sommet de $\mathtt{p}_1$ est strictement supérieur au sommet de $\mathtt{p}_2$

Le but est de répartir les éléments de la pile `p` dans les piles $\mathtt{p}_1$ et $\mathtt{p}_2$ en conservant ces restrictions. Quand la pile `p` est vide, il suffit de réempiler les éléments de $\mathtt{p}_2$ sur la pile $\mathtt{p}_1$ pour obtenir une pile triée dans l'ordre décroissant.
On va comparer $s$, le sommet de `p`, aux sommets $s_1$ et $s_2$ de $\mathtt{p}_1$ et $\mathtt{p}_2$ :

- si $s = s_1$, on empile $s$ sur $\mathtt{p}_1$
- si $s = s_2$, on empile $s$ sur $\mathtt{p}_2$
- si $s_1 > s > s_2$, on empile $s$ sur $\mathtt{p}_1$ (par exemple)
- si $s_1 < s$, on va dépiler $\mathtt{p}_1$, et empiler les éléments dans $\mathtt{p}_2$ au fur et à mesure, jusqu'à ce que le sommet de $\mathtt{p}_1$ devienne $\geqslant s$ ; il ne restera plus qu'à empiler $s$ sur $\mathtt{p}_1$. Il faut cependant faire attention à ne pas chercher à dépiler $\mathtt{p}_1$ si elle est déjà vide : dans ce cas, $s$ est strictement inférieur à tous les éléments de $\mathtt{p}_1$, on va donc « vider » $\mathtt{p}_1$ dans $\mathtt{p}_2$ et réinitialiser $\mathtt{p}_1$ en lui empilant l'élément $s$.
- On fera de façon analogue si $s_2 > s$

1. Coder cet algorithme en Python.
2. À quelle méthode de tri correspond cet algorithme ?