❑ **Exercice 1** : *Inversion au sommet*
On suppose qu'on dispose d'une structure de données de type pile dotée de son interface habituelle c'est à dire `empiler`, `dépiler` et `est_vide`. Proposer une suite d'opérations permettant d'inverser, lorsqu'ils existent, les deux éléments situés au sommet de cette pile. Si la pile contient moins de deux éléments, elle doit rester en l'état.

❑ **Exercice 2** : *Un exemple de complexité amortie*
On prend l'exemple de la structure de données implémentant le type de *list* de Python grâce à un tableau dynamique en C (voir TP). On considère un tableau donc la taille initiale est 1 et sur lequel on effectue n opérations `append`. Le but de l'exercice est de montrer qu'on obtiendra alors un nombre d'opérations en $O(n)$. On dira alors que le `append` a une **complexité amortie** en $O(1)$.

1. Montrer que le coût du redimensionnement d'un tableau de taille n est un $O(n)$.
2. Montrer que n opérations `append` vont nécessiter $\lfloor \log_2(n) \rfloor$ redimensionnement de tableau.
3. Donner la taille des tableaux lors de chaque redimensionnement.
4. En déduire le coût des redimensionnements.
5. Montrer que le coût total est un $O(n)$.

❑ **Exercice 3** : *Implémentation d'une file avec deux piles* On reprend l'implémentation d'une file avec deux piles (voir TP), le but de l'exercice est d'établir la complexité de `défiler`

1. Donner les complexités dans le pire et le meilleur des cas
2. On suppose à présent qu'en tout on a enfilé et défilé n éléments. Montrer que le nombre total d'opérations nécessaire est un $O(n)$.