

* Les exercices marqués d'une étoile sont à faire à la maison.

Une file est une structure de données *abstraite* sur laquelle sont définies trois opérations :

- `empty(F)` qui teste si la file `F` est vide ;
- `put(x,F)` qui ajoute un élément `x` en queue de la file `F` ;
- `get(F)` qui enlève l'élément en tête de la file `F` et le renvoie.

Exercice 1. *Sedgewick.*

Dans la suite suivante, une lettre indique un *put* et un astérisque indique un *get*. Donnez la suite de *get* exécutés lorsqu'on applique à une file (initialement vide) la suite d'opérations indiquée :

E A S * Y * Q U E * * * S T * * * I O * N * * *

Exercice 2. *Files implémentées par de listes chaînées.*

1. Implémentez les files à l'aide d'une liste chaînée où le premier élément de la file est en tête de la liste. Quelle est la complexité des opérations `empty`, `get` et `put` ?

Pour améliorer la complexité de l'opération `put`, on se propose d'utiliser des *listes avec pointeur de queue*. Une telle liste est représentée par une structure `ListEnd` dans laquelle la structure `Cellule` est utilisée pour représenter les éléments ou les nœuds de la liste :

<pre> 1 class Cellule { 2 Object val 3 Cellule next 4 }</pre>	<pre> 1 class ListEnd { 2 Cellule first 3 Cellule last 4 }</pre>
---	--

Le champ `first` contient une référence à la première cellule, comme pour une liste ordinaire. Le champ `last` contient une référence à la dernière cellule. (Si la liste est vide, les deux champs valent `null`.)

2. Implémentez les files à l'aide de listes avec pointeur de queue.

On se propose maintenant d'implémenter les files avec deux listes ordinaires :

```

1 class List2 {
2     Cellule first
3     Cellule second
4 }
```

On enfile les éléments dans la liste `first`, et on les défile depuis la liste `second`. Lorsque `second` est vide et on veut défiler un élément, on vide la liste `first` et on la stocke en inverse dans `second`.

3. Implémentez les files à l'aide de deux listes. Combien faut-il de temps pour enfiler un élément ? Pour défiler un élément ? Pour enfiler n éléments et les défiler tous ?
4. Quels sont les avantages et les défauts de chacune de ces implémentations ?

Exercice 3. *Files implémentées par de tableaux.*

On suppose que la file a une taille maximale de N .

1. Proposez une implémentation de la file avec un tableau.

2. Quelle est la complexité en nombre d'opérations d'un **empty**, d'un **get** et d'un **put** dans votre implémentation ? Proposez, si ce n'est pas le cas, une implémentation de la file telle que le nombre d'opération pour le **get** et le **put** soit constant.
3. * Programmer en java votre impémentation. Veiller à ce que la complexité de votre implémentation des opérations **empty**, **get** et **put** soit constante.