# ❖ TAD Lineales. Exámenes.

**■ Mayo 2018)**

Extiende la implementación basada en nodos doblemente enlazados del TAD Lista con la siguiente operación: **`void repartir()`**

Dicha operación debe realizar una transformación de la lista de tal forma que: (i) delante del primer elemento de la lista original se coloquen todos los elementos que sean estrictamente menores que él (a este grupo de elementos que quedan delante del primero de la lista original lo llamaremos primer tramo); (ii) después del primer elemento de la lista original permanezcan todos los elementos que son iguales o mayores que él (a este otro grupo de elementos que quedan detrás del primero de la lista original lo llamaremos segundo tramo). Así mismo, los elementos de cada uno de los tramos tienen que preservar las posiciones relativas que tenían en la lista original (esto es, si dos elementos, X e Y , están en un mismo tramo y el elemento X aparecía antes que Y en la lista original, X seguirá apareciendo antes que Y en el tramo). Si la lista está vacía, la operación no tendrá ningún efecto.

Por ejemplo, si la lista contiene los siguientes elementos (de principio a fin):

**`8 10 5 19 3 8 4 7 2`**

tras ejecutar dicha operación los elementos de la lista, de principio a fin, serán:

**`5 3 4 7 2 8 10 19 8`**

En la implementación no deben crearse ni destruirse nodos, ni tampoco realizarse asignaciones entre los contenidos de los nodos. Aparte de implementar la operación, debes indicar la complejidad de la misma.

**■ Junio 2018)**

Queremos extender la clase **queue** (lo importante para el ejercicio es que internamente la cola está representada con una lista enlazada simple de nodos dinámicos) con una nueva operación que inserte en una lista enlazada ordenada los elementos de otra lista enlazada ordenada recibida como argumento, de tal forma que la lista resultante quede también ordenada. La lista recibida como argumento pasará a ser vacía.

Para resolver este ejercicio no se puede crear ni destruir memoria dinámica (hacer **new** o **delete**), ni tampoco modificar los valores almacenados en las listas enlazadas.

**■ Sep 2018)**

Implementa una función que dada una cola de números enteros que está ordenada crecientemente según el valor absoluto de sus elementos, la modifique de forma que quede ordenada crecientemente según el valor de sus elementos. Puedes utilizar estructuras de datos auxiliares; justifica tu elección.

Por ejemplo, si la cola contiene los elementos **`-1, 1,-3, 4, 5,-7, 9, 10,-15`**, al final debe contener los elementos **`-15,-7,-3,-1, 1, 4, 5, 9, 10.`**

**■ Sep 2018) (2)**

Queremos extender la clase **deque** (lo importante para el ejercicio es que internamente la cola está representada con una lista enlazada circular doble de nodos dinámicos con nodo fantasma) con una nueva operación engordar que añada a una lista enlazada doble el contenido de otra lista enlazada doble dada de la siguiente forma: los nodos de la segunda lista se colocarán alternativamente al principio y al final de la primera lista. La lista recibida como argumento pasará a ser vacía.

Para resolver este ejercicio no se puede crear ni destruir memoria dinámica (hacer **new** o **delete**), ni tampoco modificar los valores almacenados en las listas enlazadas.

**■ Mayo 2019)**

"Estremecer" una lista consiste en colocar todos los elementos que aparecen en posiciones pares, por orden de aparición, seguidos de todos los que aparecen en posiciones impares, por orden inverso de aparición (primero el último, después el penúltimo, etc). Por ejemplo, el resultado de "estremecer" a

```
0 1 2 3 4 5 6 7
```

es

```
0 2 4 6 7 5 3 1
```

(**Importante:** consideramos que las posiciones comienzan en 0, es decir, la posición del primer elemento es la 0, la del segundo elemento es la 1, etc.)
Añade una nueva operación mutadora:

```
void estremece()
```

a la implementación del TAD Lista basada en nodos doblemente enlazados, que "estremezca" la lista, y determina justificadamente su complejidad. Dicha operación no puede invocar, ni directa ni indirectamente, operaciones de manejo de memoria dinámica (**new** o **delete**), ni tampoco puede realizar asignaciones a los contenidos de los nodos.

**■ Junio 2019)**

Accidentes aéreos. Problema E07.