Estructuras de Datos

Grupos B y D

Convocatoria ordinaria 2022-2023

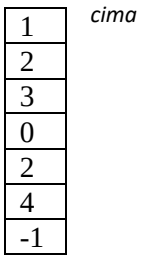
Recuerda poner tu nombre y tu usuario del juez en un comentario al principio de todos los archivos que entregues

Ejercicio 1 [2.75 puntos]

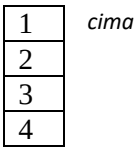
Una pila es una *pila de Hanoi* cuando cada elemento es estrictamente mayor que todos los que tiene encima. Se pide extender la implementación del TAD Pila basada en nodos enlazados con una nueva operación, *hanoificar*, que:

1. elimine todos aquellos elementos que tengan por encima elementos más grandes.
2. devuelva como resultado una nueva pila (el *resto*) con los elementos eliminados. Los elementos en *resto* deberán aparecer en el mismo orden que en la pila original.
3. tenga, como prototipo, `Pila<T> hanoificar()`

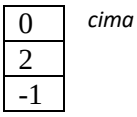
A continuación, se ilustra esta operación con un ejemplo:



Pila *P* antes de ejecutar *P*.`hanoificar()`



Pila *P* tras ejecutar *P*.`hanoificar()`



Pila devuelta por *P*.`hanoificar()` (el *resto*)

Ten en cuenta que

- La operación no podrá invocar, ni directa, ni indirectamente, ninguna operación de manejo de memoria dinámica (*new*, *delete*). Tampoco podrá asignar valores a los contenidos de los nodos, ni realizar copias de dichos valores en otras variables.
- La operación deberá ser lo más eficiente posible.

Además de implementar esta operación, deberás determinar justificadamente su complejidad.

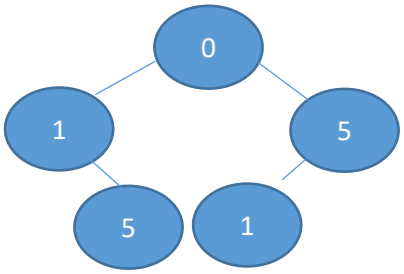
Ejercicio 2 [2.75 puntos]

Un árbol de enteros es *rebuscado* cuando, bien es vacío, o bien no es vacío y cumple las siguientes condiciones:

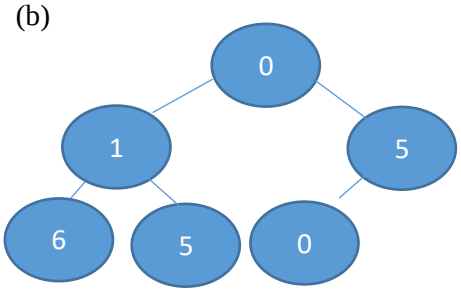
1. El mínimo de todos sus valores ocurre una única vez en el árbol.
2. Tanto el hijo izquierdo como el hijo derecho son, a su vez, *rebuscados*.

Por ejemplo, el árbol (a) en la siguiente figura es *rebuscado*, mientras que el árbol (b) no lo es:

(a)



(b)



Desarrolla un algoritmo eficiente que, tomando como entrada un árbol binario, devuelta *true* si el árbol es *rebuscado*, y *false* en caso contrario. El punto de entrada a este algoritmo será la función:

```
bool es_rebuscado(const Arbin<int>& a).
```

Aparte de implementar este algoritmo, debes determinar justificadamente su complejidad.

Ejercicio 3 [4.5 puntos]

Nos han encargado implementar un sistema para la gestión de una biblioteca. Cada libro tiene una signatura identificativa. Y de cada libro hay un cierto número de ejemplares en la estantería que pueden ser prestados. Cada socio de la biblioteca tiene un código de identificación, y puede llevarse prestados libros (no más de un ejemplar de cada libro) que deben devolverse en una fecha indicada al realizar el préstamo. Cuando todos los ejemplares del libro que solicita un socio están prestados, automáticamente el socio se incluye en la lista de espera de solicitantes de préstamo de ese libro.

Para llevar a cabo la implementación de este sistema hemos decidido desarrollar un TAD `Biblioteca` con las siguientes operaciones:

- `Biblioteca()`: Operación constructora que crea un sistema de gestión de biblioteca vacío.
- `annadir_libro(signatura, num_ejemplares)`: Añade un nuevo libro al sistema, con signatura `signatura` y con `num_ejemplares` ejemplares listos para ser prestados. Si ya está dado de alta en el sistema un libro con dicha signatura, la operación lanzará una excepción `ELibroExistente`.
- `annadir_socio(id, nombre)`: Añade un nuevo socio al sistema, con código de identificación `id` y nombre `nombre`. En el momento de añadir el socio al sistema no cuenta con libros prestados. Si ya existe un socio con código de identificación `id`, la operación lanzará una excepción `ESocioExistente`.
- `prestar(signatura, id, fecha)`→`int`: Permite prestar el libro con signatura `signatura` al socio con código de identificación `id` siendo `fecha` la fecha en la que deberá devolverse, o colocar al socio al final de la lista de espera del libro en caso de que todas los ejemplares disponibles estén ya prestados. Si no existe un libro con esa signatura o no existe un socio con ese código de identificación la operación elevará una excepción `EPrestamoNoAdmitido`. Si socio y libro existen, pero el socio ya tiene prestado un ejemplar del libro con esa signatura, el préstamo no podrá realizarse y se devolverá `0`, no llevándose a cabo ninguna acción. Si socio y libro existen y el socio no tiene prestado un ejemplar del libro con esa signatura: (a) el préstamo podrá realizarse si no están todos los ejemplares del libro prestados, quedando en ese caso registrado el préstamo con la fecha indicada y se devolverá `1`; o, (b) si estuviesen prestados todos los ejemplares, se colocará al socio al final de la lista de espera del libro y se devolverá `2`.
- `primeroEnEspera(signatura)-><codigo+nombre>`: Devuelve el código y el nombre del primer socio que está esperando para poder conseguir tener prestado un ejemplar del libro con signatura `signatura`. Si no existe un libro con signatura `signatura`, la operación elevará una excepción `ELibroInexistente`. Si no hay nadie en espera, la operación elevará una excepción `ESinEsperas`.
- `prestados(id)`→`lista<signatura+fecha>`: Devuelve una lista con las signaturas y fechas de devolución de los libros que tiene prestados el socio con código de identificación `id`. La lista estará ordenada crecientemente por signatura de los libros prestados. La lista estará vacía si el socio no tiene ningún libro prestado. Si no existe un socio con código de identificación `id`, la operación elevará una excepción `ESocioInexistente`.
- `devolver(signatura, id, fecha)`→`bool`: Permite al socio con código de identificación `id` devolver el libro con signatura `signatura`. Si no existe un libro con esa signatura o no existe un socio con ese código de identificación la operación elevará una excepción `EDevolucionNoAdmitida`. Si socio y libro existen pero el socio no tiene prestado ese libro, devolverá *false* y no se llevará a cabo ninguna acción. Si socio y libro existen y el socio tiene prestado ese libro, se eliminará el registro de dicho préstamo y devolverá *true*. En caso de haber algún socio en la lista de espera del libro, automáticamente se prestará el ejemplar devuelto al primer socio de la lista usando `fecha` como fecha de devolución del préstamo realizado automáticamente. Esa aparición del socio en la lista de espera debe ser eliminada de la lista.

Ten en cuenta que la implementación de las operaciones debe ser lo más eficiente posible. Por tanto, debes elegir una representación adecuada para el TAD. Además, debes implementar las operaciones y justificar la complejidad de cada una de ellas.