Nombre:	LU:	Comisión:	Cant. hojas:

Estructuras de Datos - Primer Parcial

Licenciatura en Ciencias de la Computación – Ingeniería en Computación – Ingeniería en Sistemas de Software
Universidad Nacional del Sur – 30/9/2023

Observaciones generales:

- **REALICE LOS EJERCICIOS EN HOJAS SEPARADAS**. □ □ □ □
- Lea todo el ejercicio antes de comenzar a desarrollarlo.
- Numere y ponga su nombre a todas las hojas. Indique cuántas hojas entrega. □ □ □ □
- Recuerde que se evalúan correctitud, eficiencia y legibilidad de sus soluciones.

Importante: para resolver este examen utilice las interfaces presentadas en clase. Al final del examen se encuentra un recordatorio de los métodos que cada una provee.

Ejercicio 1:

Suponga que cuenta con la clase listaDE<E> que implementa la interfaz PositionList<E> vista en clase. Esta clase implementa una lista doblemente enlazada con enlaces al primer y último elemento. Agregue un método a esta clase con la siguiente signatura: `public int superEliminar(E elem) throws EmptyListException`. Esté método deberá eliminar todas las apariciones (comparar por equivalencia) del elemento elem en la lista receptora del mensaje. Retorna la cantidad de elementos eliminados. Deberá lanzar la excepción correspondiente en caso de que la lista esté vacía. Si utiliza otros métodos de la clase listaDE<E> deberá implementarlos.

```
public class ListaDE<E> implements PositionList<E>{
 protected Nodo<E> head;
 protected Nodo<E> tail;
 protected int cantElems;
 …
```

Ejercicio 2:

Escriba en Java un método tal que dadas dos listas genéricas L1 y L2 verifique si todos los elementos de L1 están en L2. Asuma que cuenta con la implementación del TDALista completa. Compare los elementos por equivalencia.

Ejercicio 3:

Escriba en Java un método llamado “deListaAcola” tal que, dada una lista de caracteres, retorne una cola de caracteres con los elementos de la lista en el mismo orden. Para resolver este ejercicio asuma que cuenta con la implementación del TDALista y TDACola completas. No debe modificar el estado interno de la lista recibida por parámetro.

PositionList<E>:	**Stack<E>**	**Queue<E>**
size( )	size()	size()
isEmpty()	isEmpty()	isEmpty()
first()	pop()	dequeue()
last()	push()	enqueue()
next(p)	top()	front()
prev(p)		
addFirst(e)		
addLast(e)		
addAfter(p, e)		
addBefore(p, e)		
remove(p)		
set (p, e)		
iterator()		
positions()		