Estructuras de Datos - Primer Parcial

Licenciatura en Ciencias de la Computación – Ingeniería en Computación – Ingeniería en Sistemas de Software
Universidad Nacional del Sur – 2/5/2024

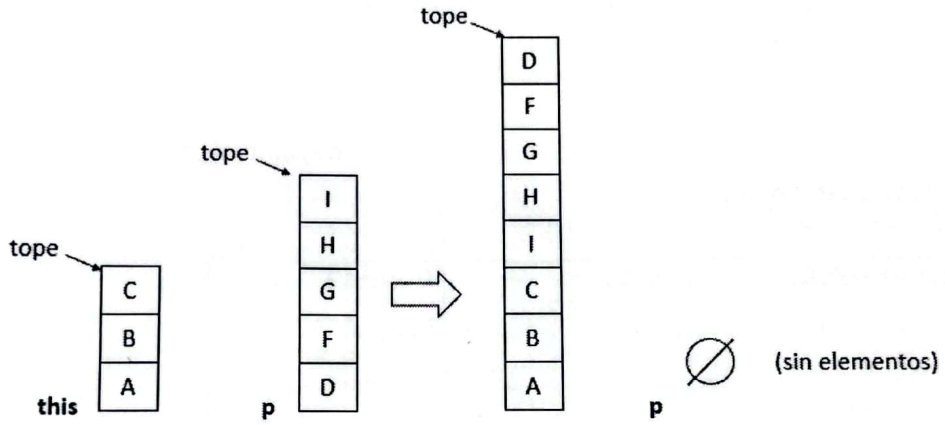
Observaciones generales:

- **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:

a) Suponga que cuenta con la clase PilaConArreglo<E> que implementa la interface Stack<E> vista en clase utilizando un arreglo. Agregue un método a la clase PilaConArreglo<E> que reciba otra pila *p* y agregue los elementos de *p* en la pila receptora del mensaje en el tope de la misma, por ejemplo:



Si utiliza el método push para apilar en la pila que recibe el mensaje deberá implementarlo. Debe considerar el caso en el que la cantidad de elementos de la pila *p* sea mayor a la cantidad de componentes disponibles en el arreglo receptor del mensaje. Recuerde que para resolver este ejercicio tiene total acceso a la estructura de la pila que recibe el mensaje.

```
public class PilaConArreglo<E> implements Stack<E>{
 protected E[] p;
 protected int tope;
 ...
```

b) Calcule el orden del tiempo de ejecución de su solución. Justifique adecuadamente.

Ejercicio 2:

a) Escriba un método con la siguiente signatura: public int maximoEnCola(Queue<Integer> q). Debe retornar el mayor valor que se encuentre en *q*. Asuma que los valores son positivos. Al finalizar el método, el contenido de *q* debe ser exactamente el mismo que antes de iniciarlo. Resuelva este ejercicio únicamente en término de los TDAs Pila y Cola. Utilice las estructuras auxiliares que crea necesarias. Asuma que cuenta con los TDAPila y TDACola totalmente implementados. Si utiliza métodos auxiliares deberá implementarlos.

b) Indique el orden del tiempo de ejecución de su solución. Justifique adecuadamente.

Ejercicio 3:

a) Escriba un método método con la siguiente signatura: public PositionList<Character> soloVocales (PositionList<Character> pl, int n). Debe retornar una nueva lista que contenga las primeras *n* vocales que se encuentren en la lista *pl*, en el mismo orden.

Por ejemplo:

Si pl = ['h', 'o', 'l', 'a', '_', 'm', 'u', 'n', 'd', 'o'] y n = 3 entonces el método deberá devolver ['o', 'a', 'u'].
Si pl = ['e', 'j', 'e', 'm', 'p', 'l', 'o'] y n = 2 entonces el método deberá devolver ['e', 'e'].
Si pl = ['s', 'k', 'y'] y n = 1 entonces el método deberá devolver [].
Si pl = ['j', 'a', 'v', 'a'] y n = 0 entonces el método deberá devolver [].

b) Indique el orden del tiempo de ejecución de su solución. Justifique adecuadamente asumiendo que las operaciones del TDA lista funcionan en orden 1.

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