Nombre:	LU:	Comisión:	Cant. hojas:

Estructuras de Datos - Segundo Parcial

Licenciatura en Ciencias de la Computación – Ingeniería en Computación – Ingeniería en Sistemas de Software
Universidad Nacional del Sur – 11/11/2023

Observaciones generales:

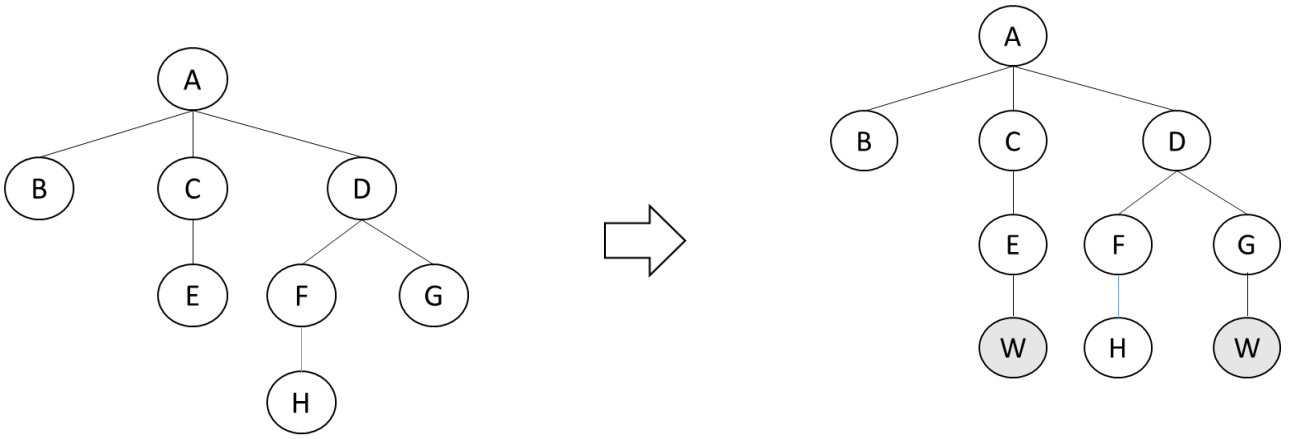
- **REALICE LOS EJERCICIOS EN HOJAS SEPARADAS**. ⇐
- Lea todo el ejercicio antes de comenzar a desarrollarlo.
- Numere y ponga su nombre a todas las hojas. Indique cuántas hojas entrega. ⇐
- Recuerde que se evalúan correctitud, eficiencia y legibilidad de sus soluciones.

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. No debe implementar en ningún momento la clase nodo ni las excepciones. Asuma que siempre que cuenta con los TDALista y TDAMapeo totalmente implementados.

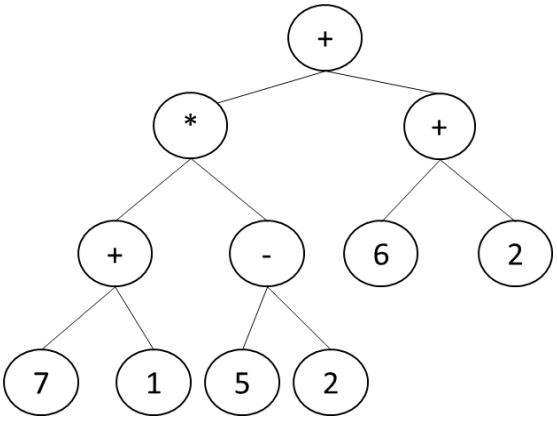
Ejercicio 1: Suponiendo que cuenta con una clase *Arbol<E>* que implementa la interfaz *Tree<E>* vista en clase. Esta clase utiliza la implementación de lista de hijos y enlace al padre. Agregue un método a esta clase cuya signatura sea `public int sizeSubarbol(Position<E> p) throws InvalidPositionException`. Este método deberá retornar el tamaño del subárbol con raíz p del árbol receptor del mensaje. Si utiliza otros métodos del TDAArbol deberá implementarlos.

```
public class Arbol<E> implements Tree<E>{
      protected int size;
      protected TNodo<E> root;
      …}
```

Ejercicio 2: Dado un árbol de caracteres *A*, un entero *x* representando un nivel del árbol y un carácter *c*, escriba un método que inserte a *c* como hijo de cada nodo de A que esté en el nivel x y que sea hoja. Por ejemplo (x=2, c= ‘w’):



Ejercicio 3: Escriba un método cuya signatura sea: public Map<Character, Integer> cantOperadores(BinaryTree<Character> a). Este método recibe un árbol binario a, que representa una expresión aritmética, y retorna un mapeo cuyas claves son las operaciones aritméticas (‘+’, ’-’, ’*’, ’/’) cuyos valores asociados corresponden a la cantidad de veces que aparece el operador en a. Recuerde que en un árbol binario de expresión los operadores se encuentran en los nodos internos y los operandos en las hojas. Por ejemplo, para el árbol de la figura el mapeo resultante debería ser: {(‘+’, 2), (‘-’, 1), (‘*’, 1), (‘/’, 0)}. Recuerde que la interfaz BinaryTree vista en clase extiende a la interfaz GTTee.



PositionList<E>:	**Tree<E>**	**BinaryTree<E>**	**Map<K,V>:**	**GTTree<E>**
size( )	size()	left(p)	size()	size()
isEmpty()	isEmpty()	right(p)	isEmpty()	isEmpty()
first()	iterator()	hasLeft(p)	get(k)	iterator()
last()	positions()	hasRigth(p)	put(k, v)	positions()
next(p)	replace(v, e)	createRoot(r)	remove(k)	replace(v, e)
prev(p)	root()	addLeft(r,p)	keys()	root()
addFirst(e)	parent(v)	addRigth(r,p)	values()	parent(v)
addLast(e)	children(v)	remove(p)	entries()	children(v)
addAfter(p, e)	isInternal(v)	attach(p,t1,t2)		isInternal(v)
addBefore(p, e)	isExternal(v)			isExternal(v)
reove(p)	isRoot(v)			isRoot(v)
set (p, e)	…			
iterator()				
positions()				