Nombre y Apellido: ....…… N° Legajo: ……

Primer Parcial de Estructuras de Datos y Algoritmos 72.34

Primer Cuatrimestre de 2014 – 22/04/2014

Ejercicio 1	*Ejercicio 2*	*Ejercicio 3*	*Nota*

Condición Mínima de Aprobación: Tener por lo menos dos ejercicios con B-

Consideraciones a tener en cuenta. MUY IMPORTANTE

- **El ejercicio que no respete estrictamente el enunciado será anulado.**
- **Se puede entregar el examen escrito en lápiz.**
- **Se tendrán en cuenta la eficiencia y el estilo de programación.**
- **Los teléfonos celulares deben estar apagados.**
- **Entregar los ejercicios en hojas separadas**

Ejercicio 1

Dada la siguiente implementación de árboles binarios

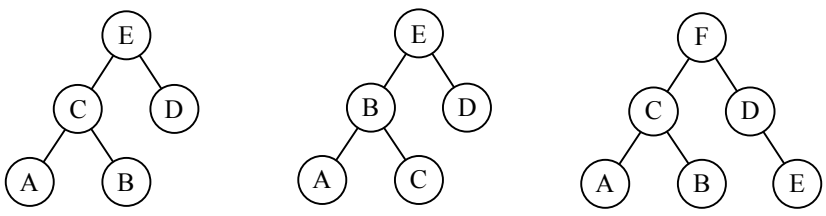
```
public class BinaryTree<T> {

	private T value;
	private BinaryTree<T> left;
	private BinaryTree<T> right;

	public BinaryTree(T value, BinaryTree<T> left, BinaryTree<T> right) {
		this.value = value;
		this.left = left;
		this.right = right;
	}

}
```

Implementar un método que determine si los valores almacenados en el árbol según un recorrido postorder se encuentran ordenados de menor a mayor (según el criterio de un comparador recibido por parámetro).

```
public static <T> boolean isPostOrderSorted(BinaryTree<T> tree, Comparator<? super T> c)
```

<u>El algoritmo no debe recorrer los nodos del árbol más de una vez ni debe crear estructuras auxiliares.</u>

Ejemplo: para el primer árbol el algoritmo retornaría true. Para los siguientes dos retornaría false.



Ejercicio 2

Se tiene la siguiente interfaz que representa un mapa en donde además de almacenar y consultar pares clave/valor, se permite eliminar el par clave/valor menos consultado y obtener la clave que más veces fue consultada:

```
public interface CustomMap<K, V> {

	/**
	 * Retorna el valor asociado a la clave, o null si la clave no existe.
	 */
	public V get(K key);

	/**
	 * Agrega un par clave/valor al mapa. Si la clave ya existe, la actualiza
	 * con el nuevo valor (en este caso esta operación se cuenta como un acceso más
	 * a la clave).
	 */
	public void put(K key, V value);

	/**
	 * Retorna la clave que más veces fue accedida.
	 */
	public K getMostAccessed();

	/**
	 * Elimina del mapa la clave (y valor) que menos veces fue accedida.
	 * Se pueden realizar sucesivas llamadas a este método.
	 */
	public void removeLeastAccessed();

}
```

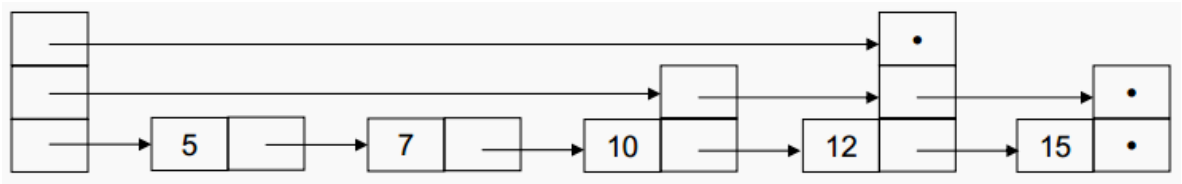
Ejemplo de uso:

```
	public static void main(String[] args) {
		CustomMapImpl<String, String> m = new CustomMapImpl<String, String>();

		System.out.println(m.get("k1"));		// Imprime null
		m.put("k1", "v1");				// Agrega k1=v1
		m.put("k2", "v2");				// Agrega k2=v2
		m.put("k3", "v3");				// Agrega k3=v3
		System.out.println(m.get("k2"));		// Imprime v2
		System.out.println(m.getMostAccessed());	// Imprime k2
		System.out.println(m.get("k1"));		// Imprime v1
		System.out.println(m.get("k2"));		// Imprime v2
		m.removeLeastAccessed();			// Elimina k3
		m.removeLeastAccessed();			// Elimina k1
		m.put("k4", "v4");				// Agrega k4=v4
		m.put("k4", "v5");				// Actualiza k4=v5
		m.put("k4", "v6");				// Actualiza k4=v6
		System.out.println(m.get("k4"));		// Imprime v6
		System.out.println(m.getMostAccessed());	// Imprime k4
		m.removeLeastAccessed();			// Elimina k2
		m.removeLeastAccessed();			// Elimina k4
	}
```

Implementar la interfaz anterior utilizando una lista lineal, teniendo en cuenta que **los métodos get y put deben tener complejidad temporal O(N), y los métodos getMostAccessed y removeLeastAccessed deben tener complejidad temporal O(1)** y no se puede utilizar la API de Java.

Ejercicio 3

Una skip-list es una lista ordenada en donde cada nodo puede mantener, además del puntero al siguiente, punteros a nodos más distantes. Además cuenta con un nodo header que no contiene información pero sí referencias a nodos.
El siguiente es un ejemplo de una lista con los valores 5, 7, 10, 12 y 15.



Se pide:
1) Definir la estructura de la clase SkipList (clases internas, propiedades)
2) Implementar el método ***belongs***, que recibe un elemento y retorna ***true*** si el mismo pertenece a la lista o ***false*** en caso contrario.