**PRIMER PARCIAL**

**INF310 SX**‒ **Estructuras de Datos II. Gestión 2-2019.**

**Subgrupo: Q-Z**

Árbol Binario

1. En un Árbol binario B (desordenado pero sin duplicados), llamamos “Racimo” a un subárbol de B, cuya raíz tiene exactamente dos hijos y ambos hijos son hojas. Sabiendo esto, en la class Arbol escriba el procedimiento

public void cutLeafBunchs(int x)

el cual elimina la hoja x, siempre y cuando x sea una hoja de un racimo del árbol. Si x no existe, o x no es una hoja de un racimo, éste procedimiento no hace nada.

**Por ejemplo**: (En el gráfico, no se dibujan los punteros null y los racimos están sombreados)

**Árbol A**

|  |  |
| --- | --- |
| **A.**cutLeafBunchs**(35);** | La hoja35 existe, pero no está en una hoja de un racimo: El árbol queda igual. |
| **A.**cutLeafBunchs**(150);** | El 150 no existe: El árbol queda igual. |
| **A.**cutLeafBunchs**(80);**  El 80 es una hoja de un racimo. Por lo tanto, es eliminado. | **Árbol A** |
| **A.**cutLeafBunchs**(10);**  El 10 es una hoja de un racimo. Por lo tanto, es eliminado. | **Árbol A** |

**Listas**

2. **JUEGO: ELIMINAR LAS TRICAS.**

En una lista se almacenan 9 números enteros (entre 1 y 10) cargados al azar, donde puede haber duplicados (pero no hay 3 iguales). El juego consiste en insertar, con el método add, un nuevo número y eliminar otro de la Lista. Si con el nuevo nro. insertado ya existen 3 con el mismo valor en la lista, se eliminan estos 3 elementos.

**public Lista()**

{**Constructor**. La lista se carga con 9 enteros aleatorios}

**public void add(int nuevoNro, int nroBotar)**

//Asuma que están entre 1 y 10 y que nuevoNro≠nroBotar

{ Si **nroBotar** **NO** existe en la Lista, este método “no hace nada”.

Se inserta **nuevoNro** a la Lista y se reemplaza por **nroBotar**. Luego, si existen 3 con el mismo valor, eliminar los nodos que

lo contienen.

}

**Por ejemplo (en el main):** Como usuario, imaginamos a la Lista P como una secuencia de pares NroPersonaje/cantVidas.

Lista P = new Lista(); //(Por ejemplo) P= [6, 5, 7, 3, 8, 9, 6, 1, 3]

P.add(**7**,3); //P = [6, 5, 7, **7**, 8, 9, 6, 1, 3] El 7 se reemplaza por el primer 3.

P.add(9, **2**); //P = [6, 5, 7, 7, 8, 9, 6, 1, 3] El número a botar **2**, no existe: La lista queda igual

P.add(7, 1); // P = [6, 5, 8, 9, 6, 3] El 7 se reemplaza por el 1. Como ya hay tres 7 en la Lista, éstos se eliminan.

P.add(**4**, 6); // P = [**4**, 5, 8, 9, 6, 3] El 4 se reemplaza por el primer 6.

P.add(**4**, 6); // P = [4, 5, 8, 9, **4**, 3] El 4 se reemplaza por el 6.

P.add(4, 8); // P = [5, 9, 3] El 4 se reemplaza por el 8. Como ya hay tres 4, éstos se eliminan.