# Francisco Alba

# Parte 1

**Importante:** Los ejercicios de esta primera parte tienen como objetivo codificar las diferentes funciones básicas necesarias para la implementar un árbol AVL.

A partir de estructuras definidas como :

**class** AVLTree:

root = **None**

**class** AVLNode:

parent = **None**

leftnode = **None**

rightnode = **None**

key = **None**

value = **None**

bf = **None**

Copiar y adaptar todas las operaciones del **binarytree.py** (i.e insert(), delete(), search(),etc) al nuevo módulo **avltree.py.** Notar que estos luego deberán ser implementados para cumplir que la propiedad de un árbol AVL

## Ejercicio 1

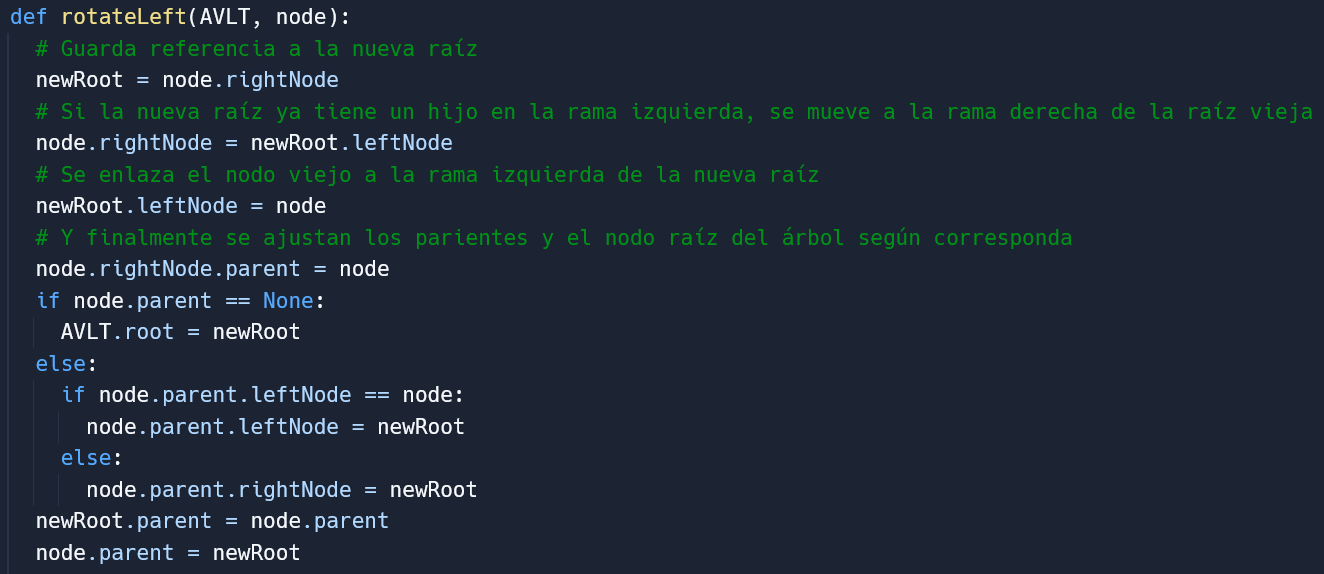
Crear un modulo de nombre **avltree.py** Implementar las siguientes funciones:

**rotateLeft(Tree,avlnode)**

**Descripción:** Implementa la operación rotación a la izquierda

**Entrada:** Un Tree junto aun AVLnode sobre el cual se va a operar la rotación a la izquierda

**Salida:** retorna la nueva raíz

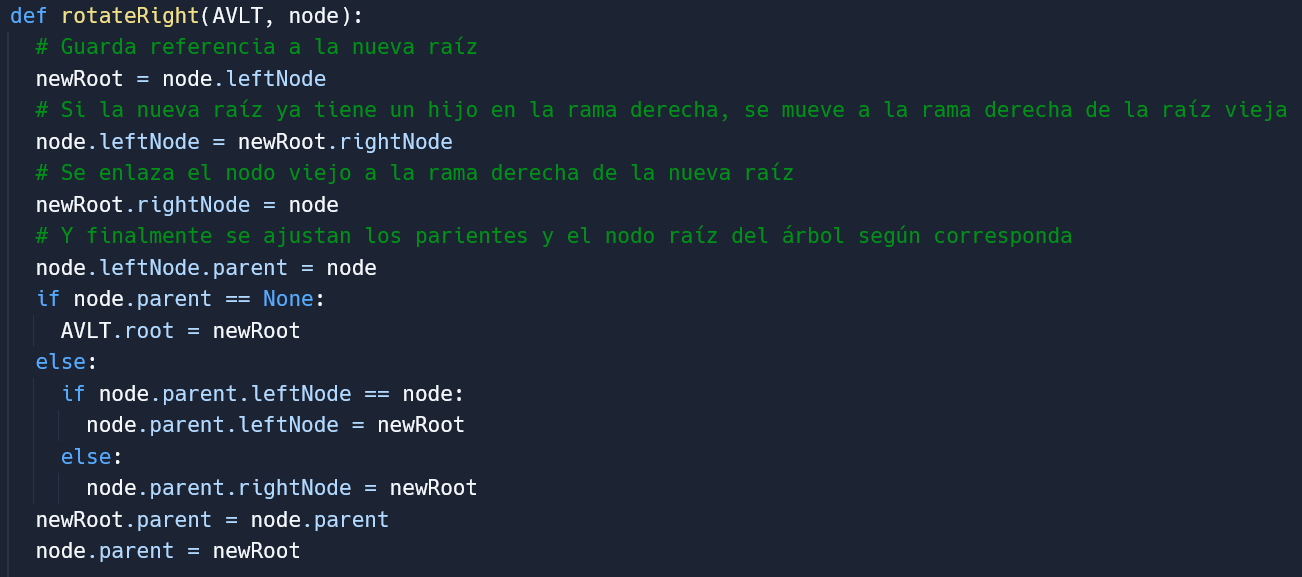


**rotateRight(Tree,avlnode)**

**Descripción:** Implementa la operación rotación a la derecha

**Entrada:** Un Tree junto aun AVLnode sobre el cual se va a operar la rotación a la derecha

**Salida: retorna la nueva raíz**



## Ejercicio 2

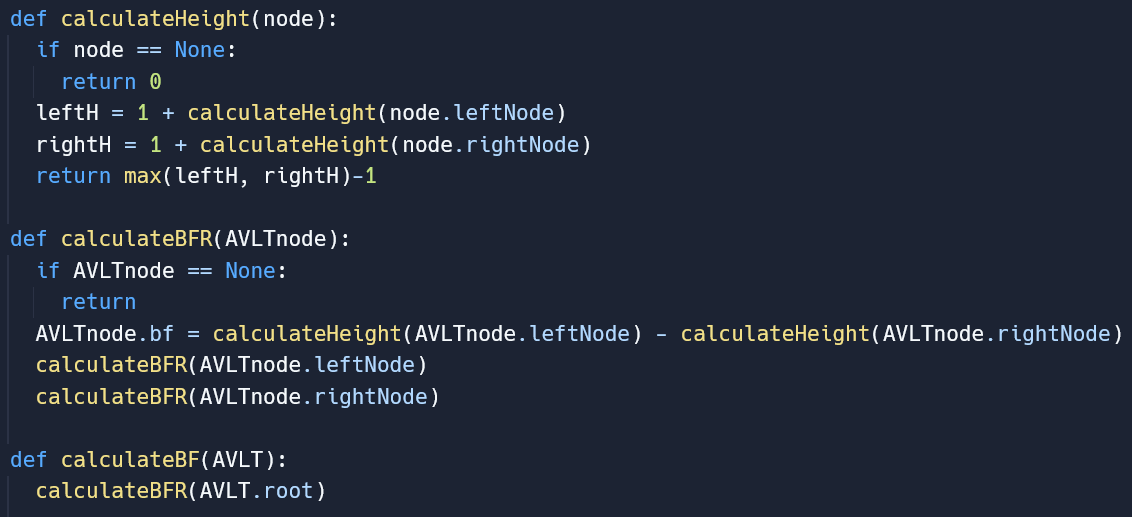
Implementar una función recursiva que calcule el elemento balanceFactor de cada subárbol siguiendo la siguiente especificación:

**calculateBalance(AVLTree)**

**Descripción:** Calcula el factor de balanceo de un árbol binario de búsqueda**.**

**Entrada:** El árbol AVL sobre el cual se quiere operar.

**Salida:** El árbol AVL con el valor de balanceFactor para cada subarbol



## Ejercicio 3

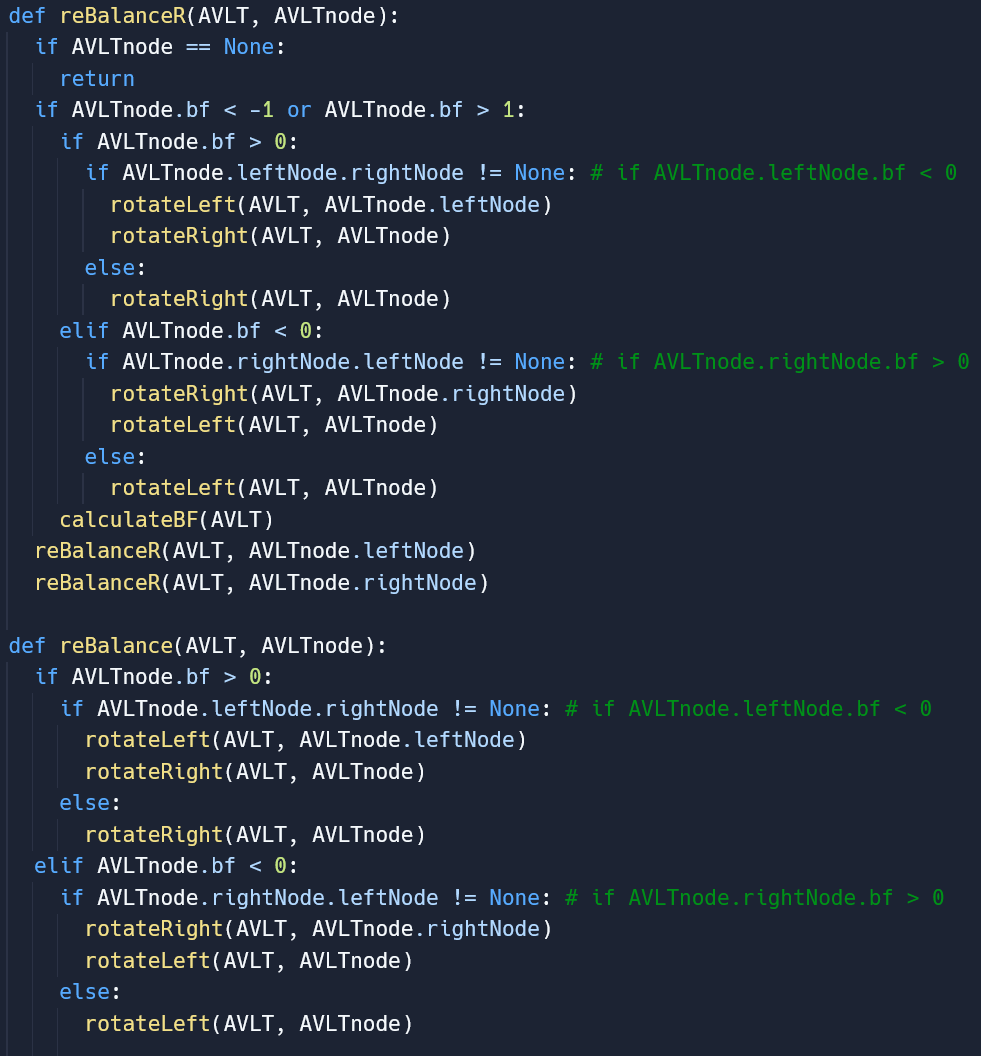
Implementar una funcion en el modulo avltree.py de acuerdo a las siguientes especifcaciones:

**reBalance(AVLTree)**

**Descripción:** balancea un árbol binario de búsqueda**.** Para esto se deberá primero calcular el **balanceFactor** del árbol y luego en función de esto aplicar la estrategia de rotación que corresponda.

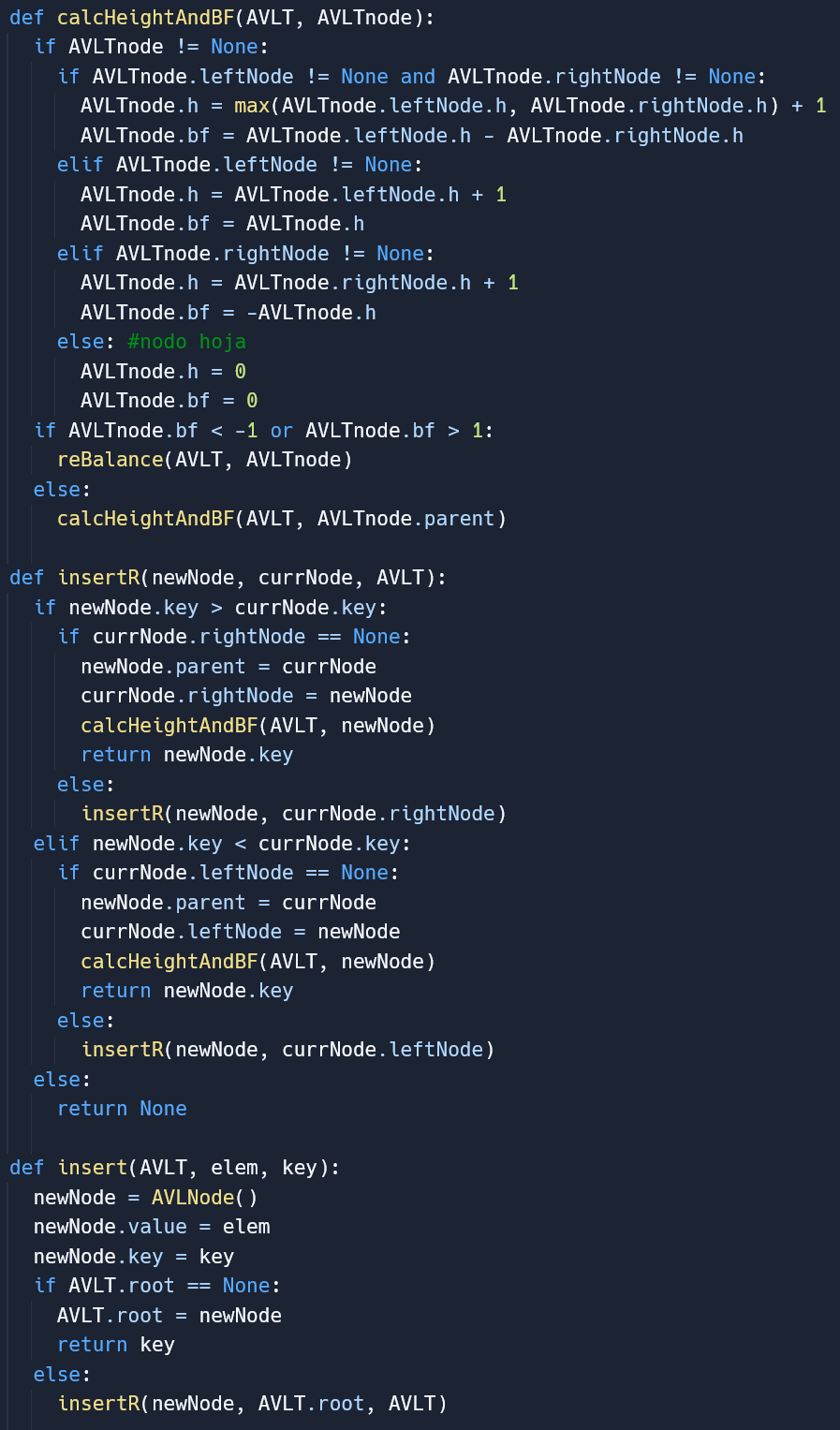
**Entrada:** El árbol binario de tipo AVL sobre el cual se quiere operar.

**Salida:** Un árbol binario de búsqueda balanceado. Es decir luego de esta operación se cumple que la altura (h) de su subárbol derecho e izquierdo difieren a lo sumo en una unidad.



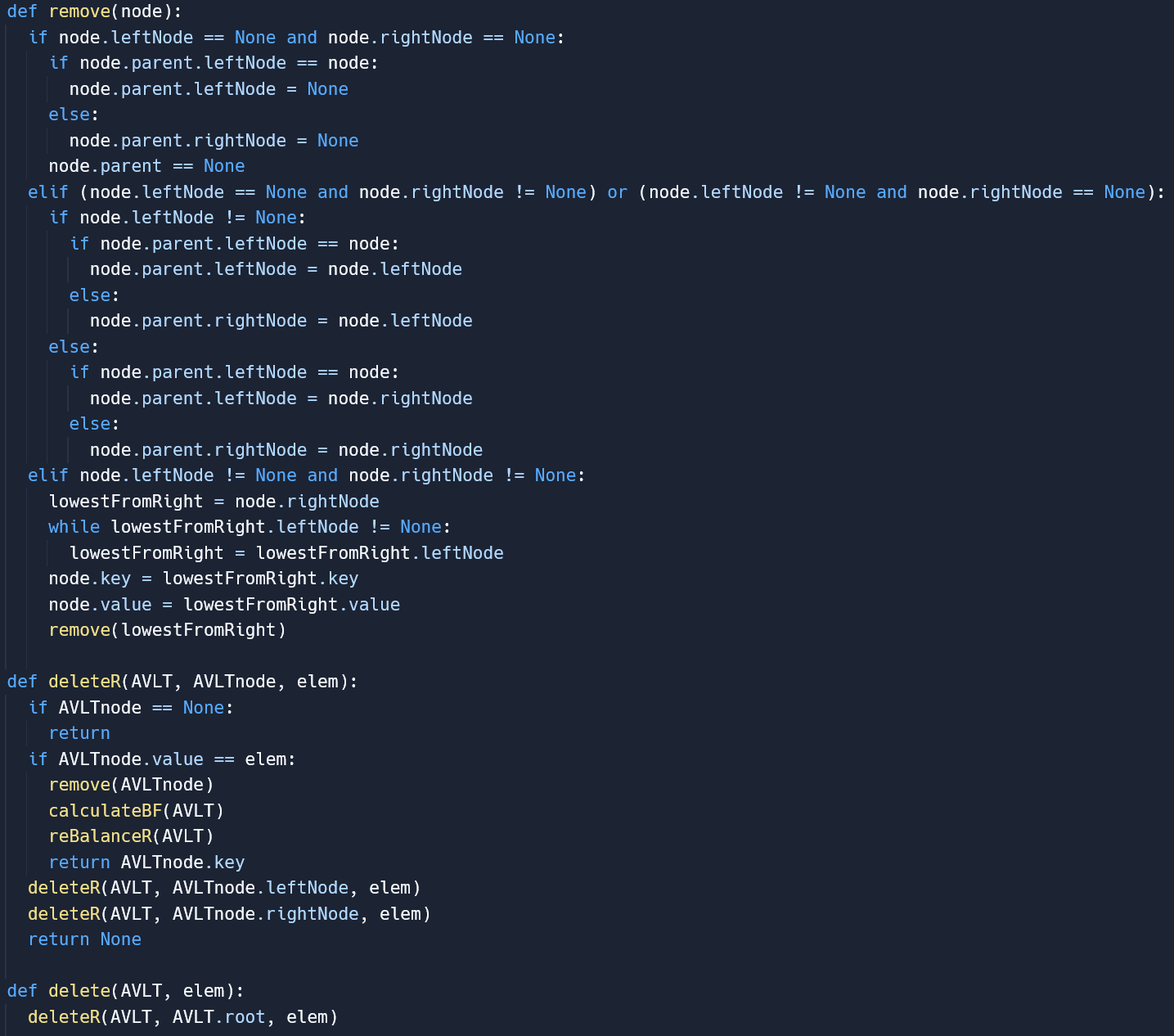
## Ejercicio 4:

Implementar la operación **insert()** en el módulo **avltree.py** garantizando que el árbol binario resultante sea un árbol AVL.



## Ejercicio 5:

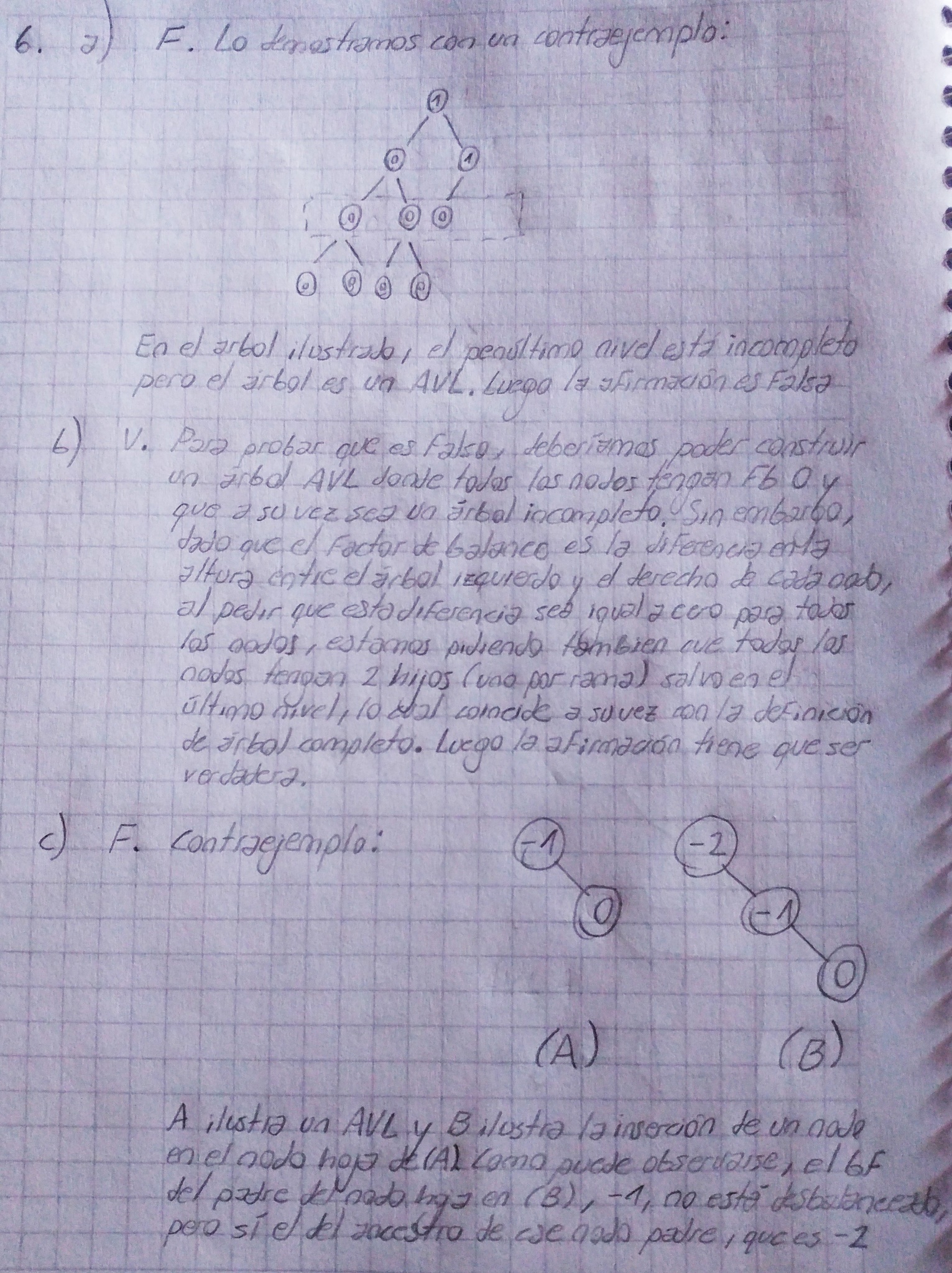
Implementar la operación **delete()** en el módulo **avltree.py** garantizando que el árbol binario resultante sea un árbol AVL.

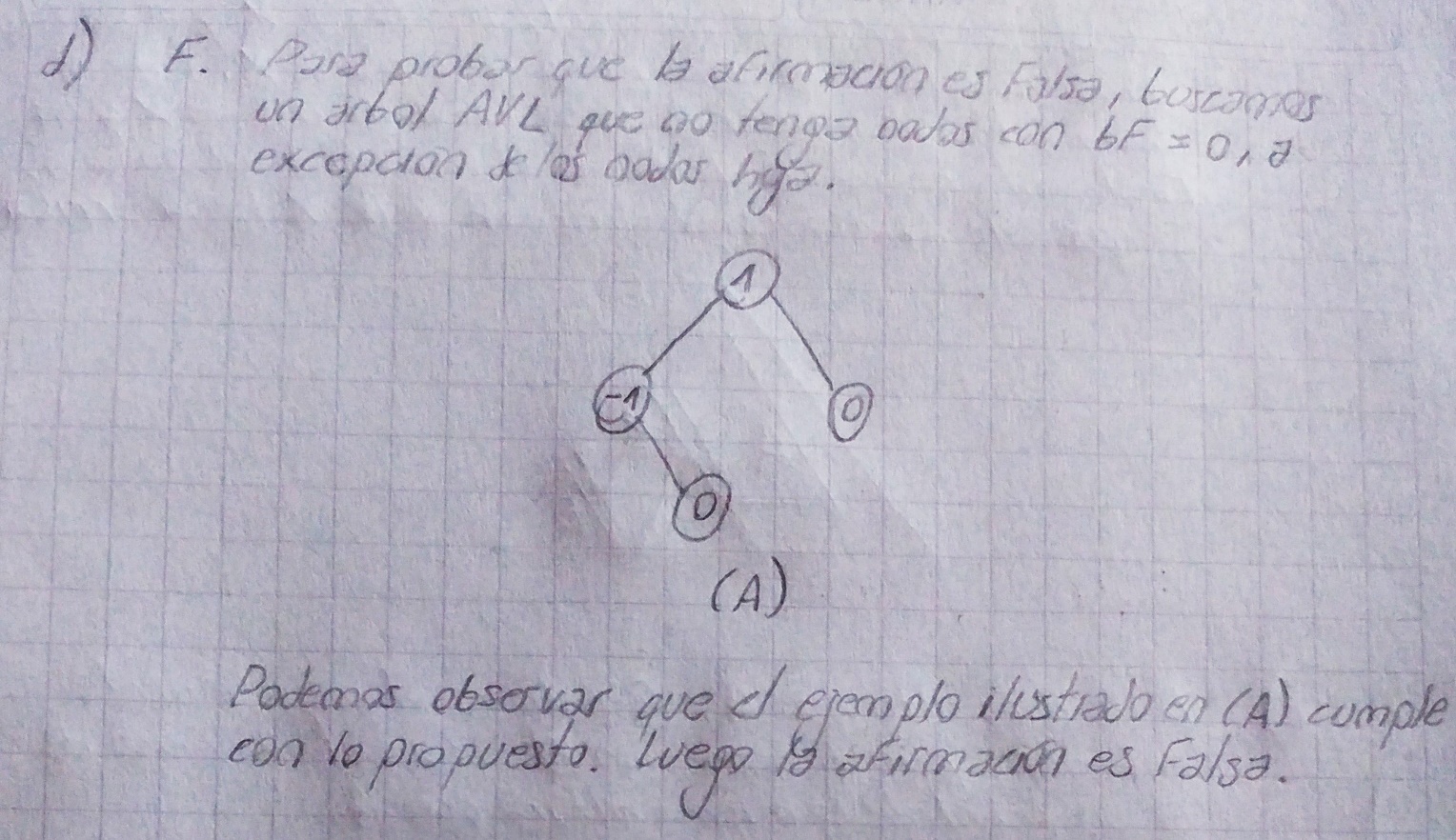


# Parte 2

## Ejercicio 6:

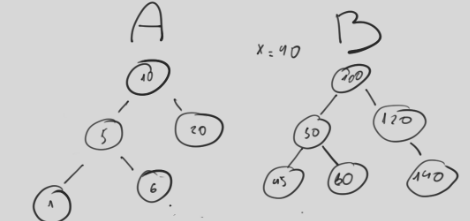
1. Responder V o F y justificar su respuesta:
   1. \_F\_ En un AVL el penúltimo nivel tiene que estar completo
   2. \_V\_ Un AVL donde todos los nodos tengan factor de balance 0 es completo
   3. \_F\_ En la inserción en un AVL, si al actualizarle el factor de balance al padre del nodo insertado éste no se desbalanceó, entonces no hay que seguir verificando hacia arriba porque no hay cambios en los factores de balance.
   4. \_F\_ En todo *AVL* existe al menos un nodo con factor de balance 0.

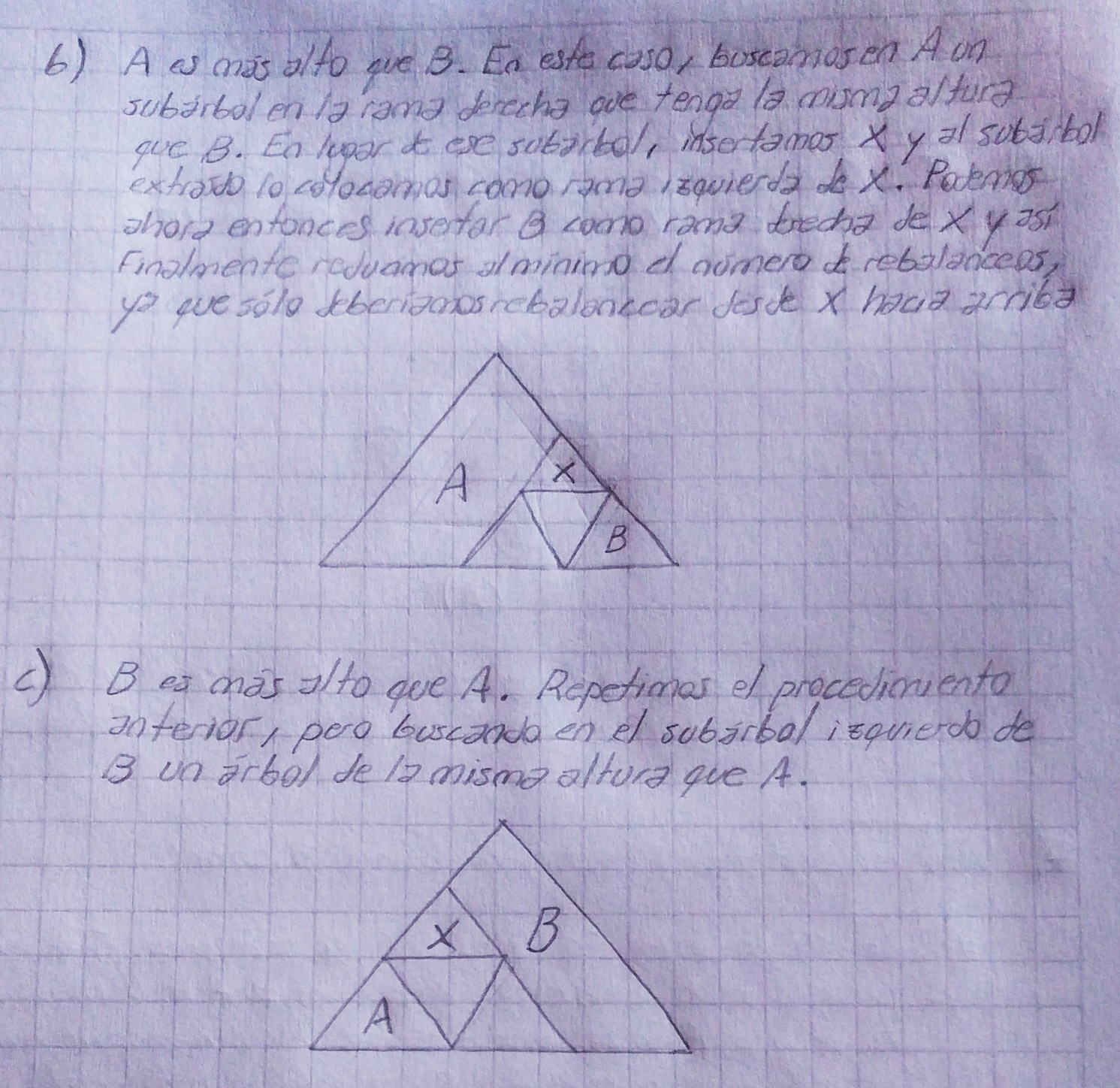
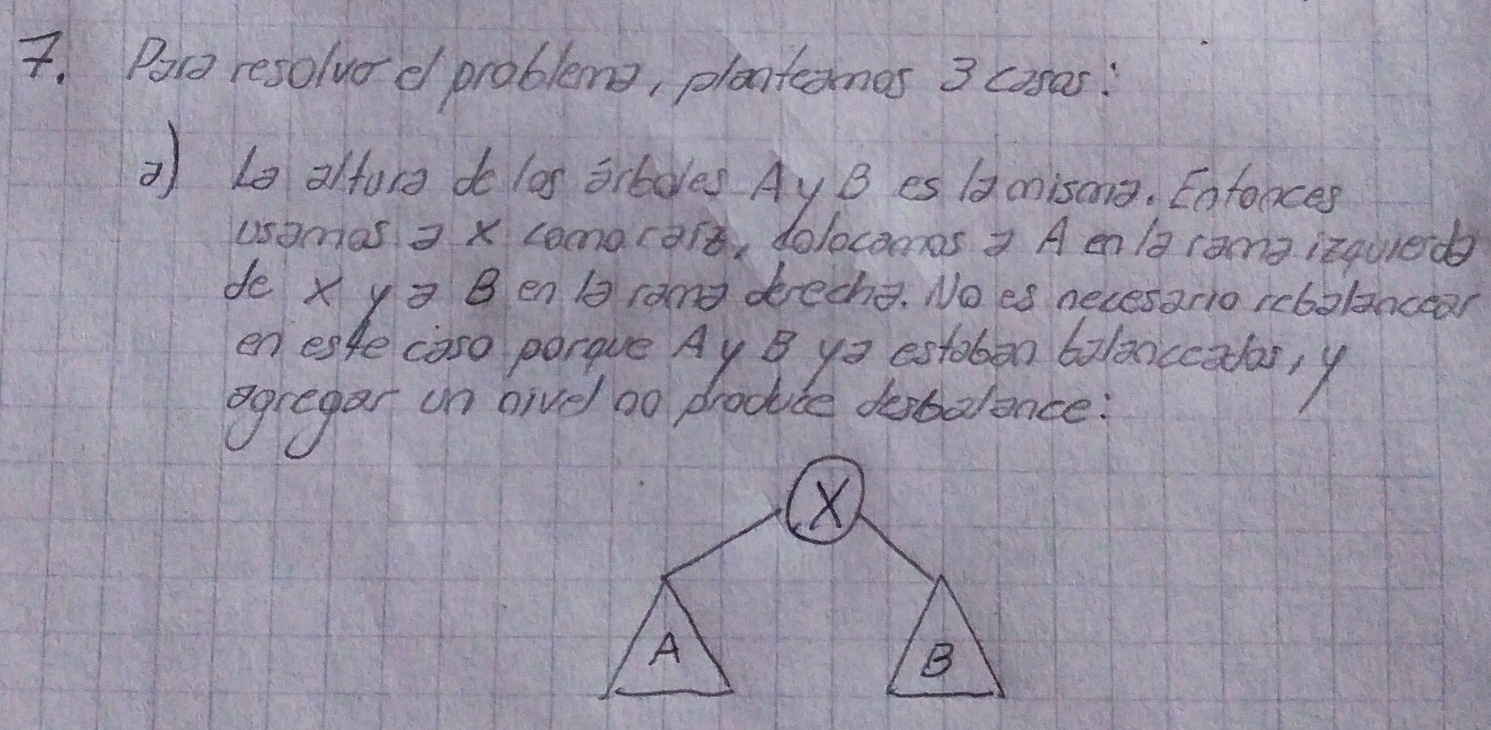




## Ejercicio 7:

Sean *A* y *B* dos *AVL* de *m* y *n* nodos respectivamente y sea *x* un key cualquiera de forma tal que para todo key *a* ∈ *A* y para todo key *b* ∈ *B* se cumple que *a* < *x* < *b*. Plantear un algoritmo *O*(log *n* + log *m*) que devuelva un *AVL* que contenga los key de *A*, el key *x* y los key de *B*.





## Ejercicio 8:

Considere una rama truncada en un AVL como un camino simple desde la raíz hacia un nodo que tenga una referencia None (que le falte algún hijo). Demuestre que la mínima longitud (cantidad de aristas) que puede tener una rama truncada en un AVL de altura h es h/2 (tomando la parte entera por abajo).

Cualquier camino desde la raíz hasta un nodo que no esté completo puede ser una rama truncada según la definición del ejercicio. Dicho nodo puede no ser necesariamente un nodo hoja.

# Suponemos que la mínima longitud de una rama truncada en un AVL de altura h es k > h/2 y buscamos llegar a una contradicción.

# Para que el AVL sea de altura h, tiene que haber una rama completa de longitud h que se conecta a la raíz.

# Como la rama truncada tiene longitud k > h/2, para no exceder la altura total del árbol h (lo cual sería una contradicción al supuesto de que el árbol es de altura h) debemos conectar la rama truncada a otra rama que tenga longitud h - k < h/2. Pero esto significa que la rama completa a la cual estamos intentando conectar la rama truncada es más corta que la rama truncada en sí, lo cual no puede ser cierto en un AVL balanceado, ya que todas las ramas de un AVL deben tener una diferencia de altura de como máximo 1.

# Por lo tanto, la suposición de que la mínima longitud de una rama truncada en un AVL de altura h es mayor que h/2 es falsa y así la mínima longitud que puede tener una rama truncada en un AVL de altura h es h/2.

# Parte 3

## Ejercicios Opcionales

1. Si n es la cantidad de nodos en un árbol AVL, implemente la operación **height()** en el módulo **avltree.py** que determine su altura en O(log n). Justifique el por qué de dicho orden.
2. Considere una modificación en el módulo **avltree.py** donde a cada nodo se le ha agregado el campo **count** que almacena el número de nodos que hay en el subárbol en el que él es raíz. Programe un algoritmo *O*(log *n*) que determine la cantidad de nodos en el árbol cuyo valor del key se encuentra en un intervalo [*a*, *b*] dado como parámetro. Explique brevemente por qué el algoritmo programado por usted tiene dicho orden.

#### A tener en cuenta:

1. **Usen lápiz y papel primero**
2. **~~No se puede utilizar otra Biblioteca mas alla de algo1.py y las bibliotecas desarrolladas durante Algoritmos y Estructuras de Datos I.~~**

## Bibliografia:

[1] Guido Tagliavini Ponce, [Balanceo de arboles y arboles AVL](https://drive.google.com/open?id=1f6uG9KWwOaOGMkDZ_fr9FWyZd8DmLGgD) (Universidad de Buenos Aires)

[2] Brad Miller and David Ranum, Luther College, [Problem Solving with Algorithms and Data Structures using Python](http://interactivepython.org/runestone/static/pythonds/index.html).