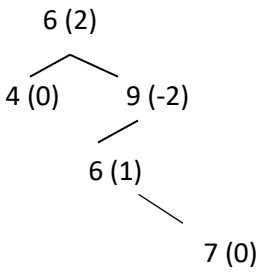
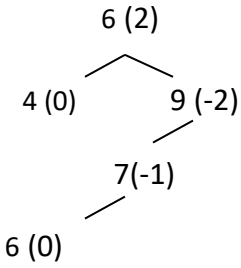
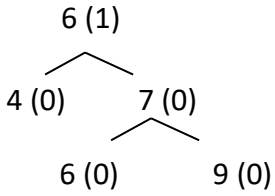
1. **Dados los siguientes árboles binarios, determinar si están balanceados y especificar el factor de balanceo para todos los nodos. En el caso de que no estén balanceados, identificar las raíces de los subárboles más pequeños no balanceados y efectuar las rotaciones necesarias para que lo estén.**

a)



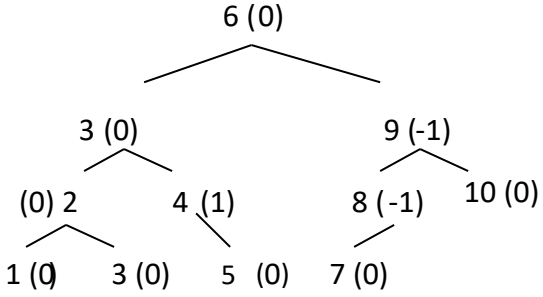
El árbol no está balanceado, los factores de balanceo están entre paréntesis, el nodo a balancear es 9. Es necesario hacer una doble rotación, *izquierda_derecha*, primero una *rotación simple a la izquierda* y después una *rotación simple a la derecha*:





rotación simple izquierda_izquierda *rotación simple derecha_derecha*

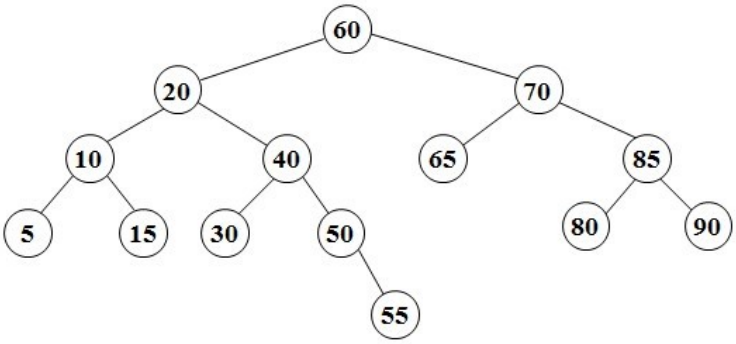
b)



El árbol está balanceado, los factores de balanceo están entre paréntesis.

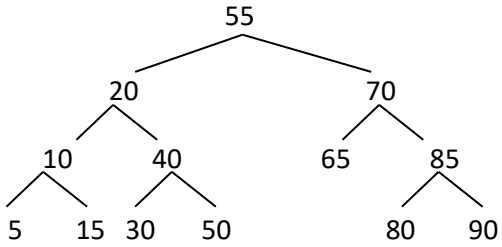
2. **Insertar los siguientes nodos en un árbol AVL vacío, indicando los pasos y rotaciones necesarias. Nodos a insertar: 10, 40, 35, 25, 60, 30, 80, 50, 27, 28, 38**

3. **Dado el siguiente árbol AVL, borrar los nodos: 60, 55, 50 y 40. Indicar los pasos y las rotaciones realizadas, en caso de que sean necesarias.**



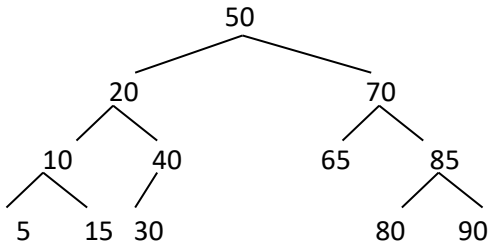
Solución:

Borrar 60 (se sustituye por el máximo del hijo izquierdo)



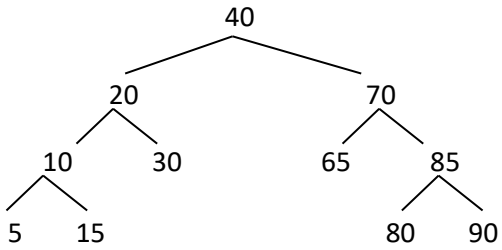
No es necesario realizar ninguna rotación.

Borrar 55 (se sustituye por el máximo del hijo izquierdo)



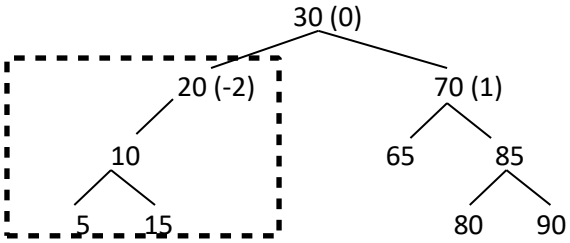
No es necesario realizar ninguna rotación

Borrar 50 (se sustituye por el máximo del hijo izquierdo)

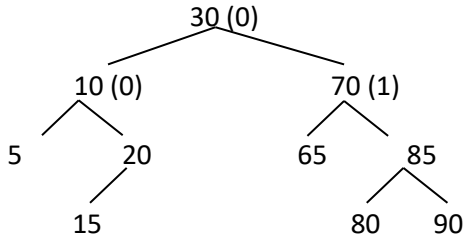


No es necesario realizar ninguna rotación

Borrar 40 (se sustituye por el máximo del hijo izquierdo)



Al borrar 30 del hijo izquierdo es necesario balancear el árbol resultante (rotación simple *izquierda_izquierda*)



4. **Extender la especificación de árbol binario con las siguientes operaciones (en pseudocódigo):**
   a. **determinar si un árbol binario es completo.**
   b. **determinar si un árbol binario es semicompleto.**
   c. **determinar si un árbol binario es un montículo de mínimos.**

5. **Indicar el contenido de un montículo de mínimos, representado por un vector, después de insertar en un montículo vacío los siguientes enteros: 4, 1, 5, 2, 9, 8, 3, 7.**
   Solución: [1,2,3,4,9,8,5,7]

Insertar 4 al final del vector: [4]
Flotar 4: [4]
Insertar 1 al final del vector: [4,1]
Flotar 1: [1, 4] 1<4? →Intercambio
Insertar 5 al final del vector: [1,4, 5]
Flotar 5: [1, 4, 5] 1<5? →No Intercambio
Insertar 2 al final del vector: [1, 4, 5, 2]
Flotar 2: [1, 2, 5, 4] 2<4? →Intercambio
Insertar 9 al final del vector: [1, 2, 5, 4, 9]
Flotar 9: [1, 2, 5, 4, 9] 2<9? →No Intercambio
Insertar 8 al final del vector: [1, 2, 5, 4, 9, 8]
Flotar 8: [1, 2, 5, 4, 9, 8] 5<8? No intercambio
Insertar 3 al final del vector: [1, 2, 5, 4, 9, 8, 3]
Flotar 3: [1, 2, 3, 4, 9, 8, 5] 5<3? Intercambio
Insertar 7 al final del vector: [1, 2, 3, 4, 9, 8, 5, 7]
Flotar 7: [1, 2, 3, 4, 9, 8, 5, 7] 4<7? No intercambio

6. **Indicar el contenido de un montículo de mínimos, representado por un vector, después de eliminar el mínimo en el montículo del ejercicio 5.**

   Solución: [2, 4, 3, 7, 9, 8, 5]

   Sustituir el mínimo (posición 1) por el último (posición máximo): [7, 2, 3, 4, 9, 8, 5]
   Hundir 7: Intercambiar 7 con el menor de sus hijos (2): [2, 7, 3, 4, 9, 8, 5]
   Intercambiar 7 con el menor de sus hijos (4): [2, 4, 3, 7, 9, 8, 5]

7. **El algoritmo de ordenación por el método del montículo (*heapsort*) inserta en un montículo todos los elementos del vector a ordenar. Después se va extrayendo sucesivamente el mínimo del montículo, de forma que los elementos quedan ordenados en orden creciente. Implementar este algoritmo de ordenación utilizando las operaciones básicas de montículo estudiadas en clase.**

   Solución:

```
tipos
vector=reg
            tamaño=0..maximo
            datos=array[1..maximo] de tipo_elemento
finreg

proc heapsort(v:vector)
                              {ordena v utilizando un vector de mínimos}
var m:montículo               {tipo y operaciones estudiadas en clase}

    Inicializar_monticulo(m)
    desde i←1 hasta v.tamaño hacer
```

```
                Insertar(v[i], m)
            findesde

            desde i←1 hasta m.tamaño hacer
                v[i]←eliminarmin(m)
            findesde
finproc
```