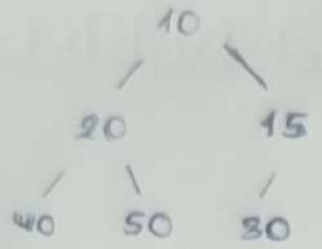
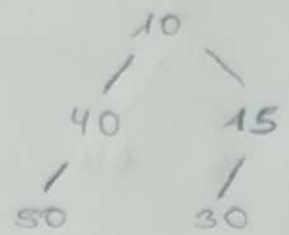
Planteamiento Practica 5

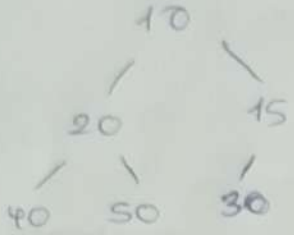
Problema 1



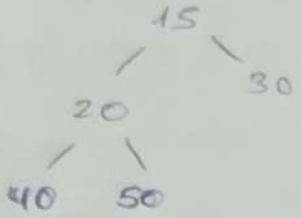
eliminar 20 →

subimos el 40 porque es el menor de los dos hijos



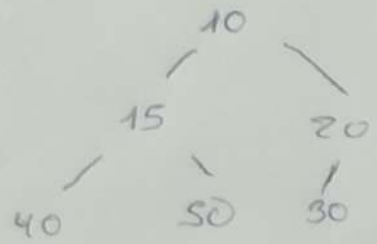


eliminar 10 →

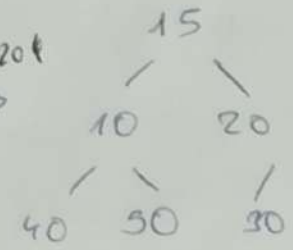


Eliminación → si es hoja se elimina sin problemas

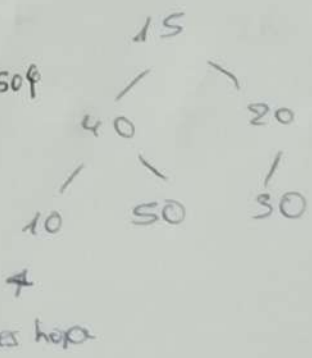
- si tiene un hijo necesitamos que ese elemento se hoja para eliminarlo por lo que la posición del elemento la ocupará el hijo
- si tiene dos hijos la posición del elemento la ocupará el hijo más pequeño así sucesivamente hasta el elemento a eliminar sea hoja.



min 15, 20 →

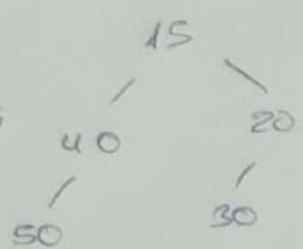


min 40, 50 →

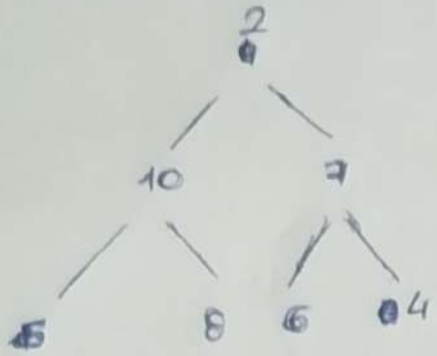


ya es hoja

→



Problema 2



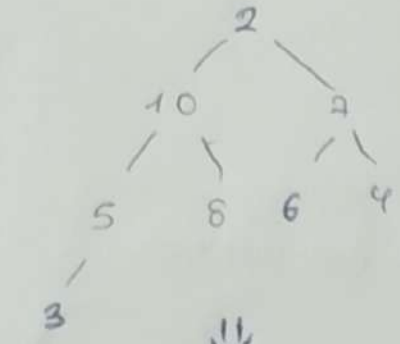
nivel par abuelo ≤ e ≤ padre

nivel impar padre ≤ e ≤ abuelo

Representación vector → | 2 | 10 | 7 | 5 | 8 | 6 | 4 |

Insertar el valor 3, nos quedaría el vector así | 2 | 10 | 7 | 5 | 8 | 6 | 4 | 3 |

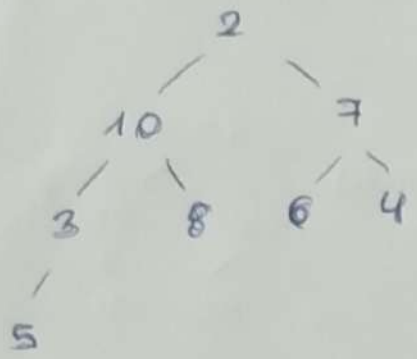
ya que lo insertamos en la primera posición libre, entonces el árbol es



el 3 está en nivel impar por lo que

padre ≤ 3 ≤ abuelo pero su padre es 5 por lo

que no se cumple por lo que lo intercambiaros



el 3 está en nivel par y cumple abuelo ≤ e ≤ padre

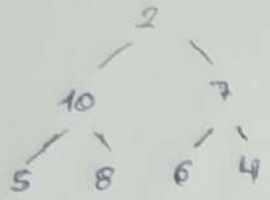
2 ≤ 3 ≤ 10 y el 5 en nivel impar y cumple

padre ≤ e ≤ abuelo 3 ≤ 5 ≤ 10

Problema 3

Siguiendo lo establecido en el problema anterior nivel por abuelo ≤ e < padre y nivel mayor padre ≤ e ≤ abuelo podemos establecer que el nivel donde se encuentra el número es en el nivel 1, esto vamos a corroborlo con el árbol anterior

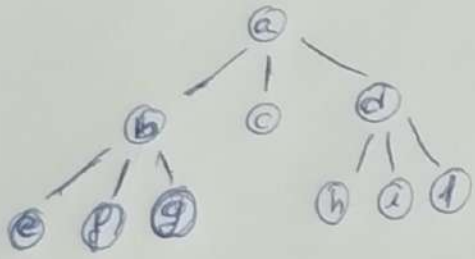
mayor = 10 y el nivel es el 1



Idea = acceder a la raíz del árbol y posteriormente a sus hijos. Ahí comprobar que hijo es el mayor y ese valor será el buscado por lo explicado anteriormente

Problema 4

Ternario → todos los nodos son hojas o tienen tres hijos



Idea = comprobar si es hoja y comprobar que si tiene hijos tiene concretamente 3 no más

Recursividad para recorrer el árbol

```
bool ternario(Agen)
    return ternario_rec(Agen, raiz)

bool ternario_rec(Agen, n)
    if (n == nulo)
    | return true
    else if (A.hijoIzqdo(n) != Nulo && A.hermDrch(n)
    !nulo)
    | esternario_rec(A.hijoIzdo(n), A)) && esternario_rec
    (A.hermDrch(n), A))
```