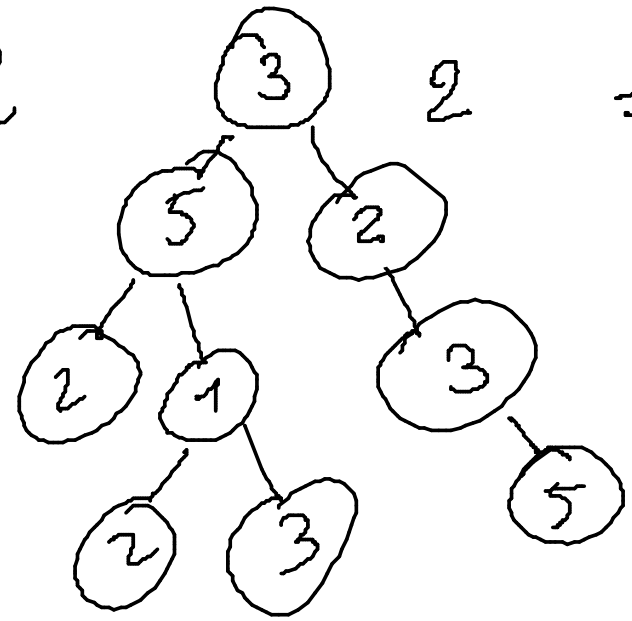
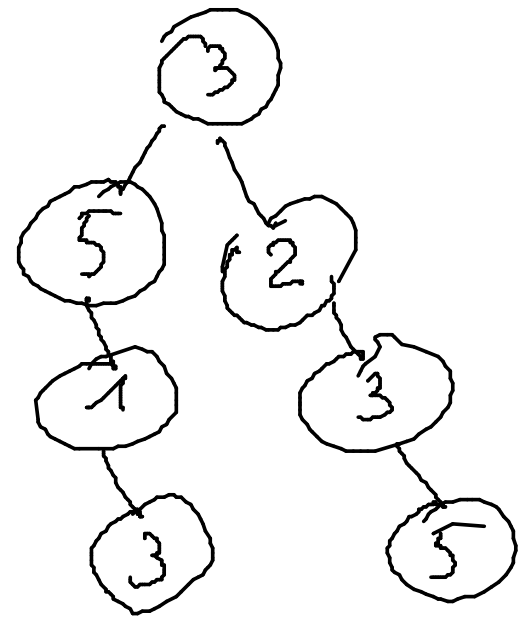
type 'a btree = Void | Node of 'a * 'a btree * 'a btree

Scrivere una funzione ricorsiva

del : 'a btree → 'a → 'a btree

che cancelli dall'albero binario tutte le foglie che contengono un valore uguale al secondo argomento

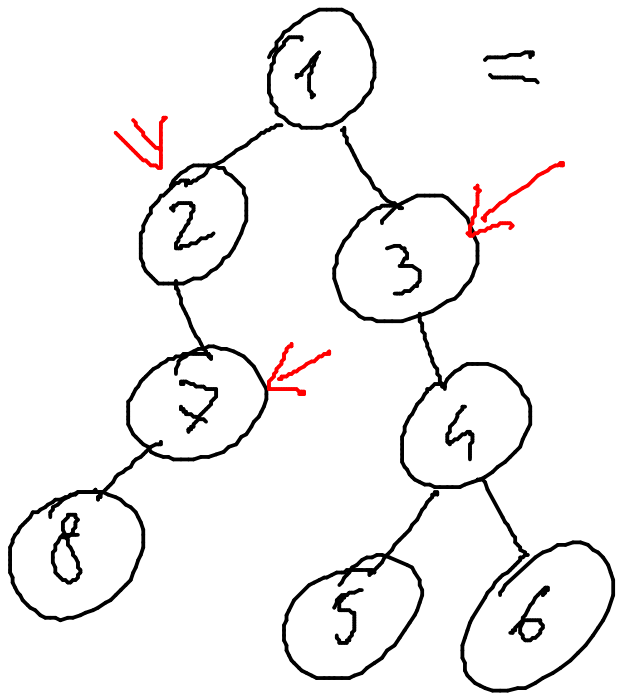
del  2 =



```
let rec del bt a = match bt with
        Void -> Void
  | Node (x, Void, Void) -> if x=a then Void
                            else bt
  | Node (x, lbt, rbt) when lbt <> Void & rbt <> Void
        -> Node (x, del lbt a, del rbt a);;
```

conte : 'a btree → int

conte bt restituisce il numero di modi di bt che hanno <u>esattamente</u> un figlio

conte 

= 3

```
let rec conte bt = match bt with
                   Void -> 0
                 | Node (x, Void, rbt) when rbt <> Void -> conte rbt + 1
                 | Node (x, lbt, Void) when lbt <> Void -> conte lbt + 1
                 | Node (x, Void, Void) -> 0
                 | Node (x, lbt, rbt) when lbt <> Void & rbt <> Void
                                        -> conte lbt + conte rbt ;;
```

Due alberi binari hanno la stessa struttura ≡

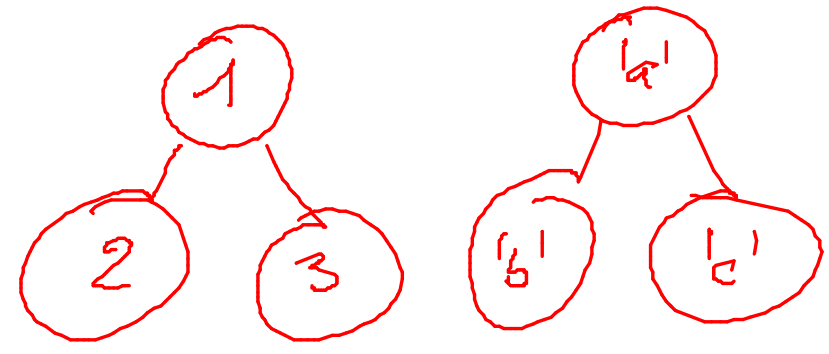
- Sono entrambi vuoti, oppure
- Sono non vuoti e i sottoalberi sinistri e destri hanno la stessa struttura.

—

sames : 'a btree → 'b btree → bool

che controlla che due alberi binari abbiano la stessa struttura

```
let rec sames bt1 bt2 = match (bt1, bt2) with
    (Void, Void) -> true
  | (Void, bt) when bt <> Void -> false
  | (bt, Void) when bt <> Void -> false
→ (Node (x, lbt1, rbt1), Node (y, lbt2, rbt2)) ->
    sames lbt1 lbt2 &
    sames rbt1 rbt2 ;;
```



cont : 'a btree → 'a list → bool

Controlla che tutti gli elementi dell'albero binario siano contenuti nelle liste

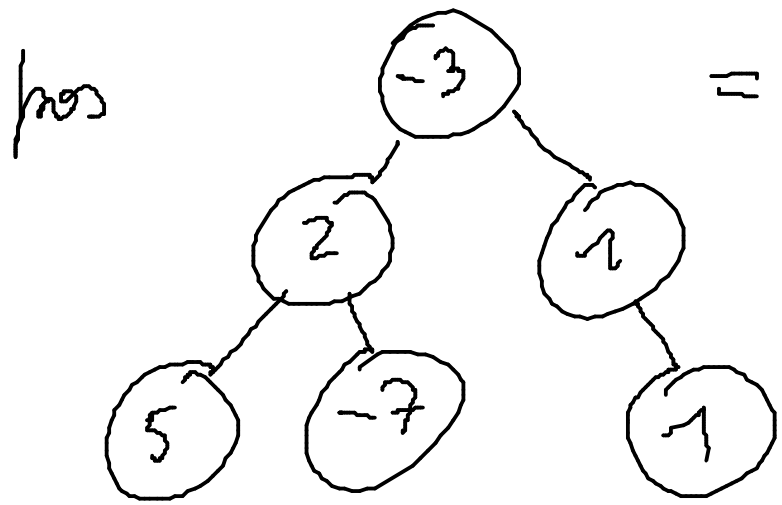
member : 'a → 'a list → bool

```
let rec cont bt l = match bt with
    Void -> true
  | Node (x, lbt, rbt) -> member x l &
                          cont lbt l &
                          cont rbt l ;;
```

```
--> if member x l
    then if cont lbt l
         then cont rbt l
         else false
    else false
```

linpos : int btree → int list

Costruisce una lista che contiene tutti e solo gli elementi pari dell'albero.

linpos 

= [2;5;-1;-1]

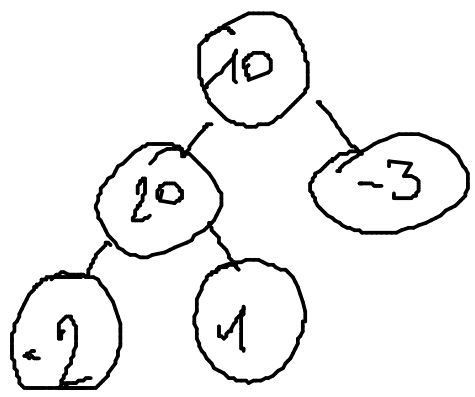
```
let rec limpos bt = match bt with
    Void -> []
  | Node (x, lbt, rbt) when x>0 ->
                x :: limpos lbt @ limpos rbt
  | Node (x, lbt, rbt) when x<=0 ->
                limpos lbt @ limpos rbt ;;
```

definire

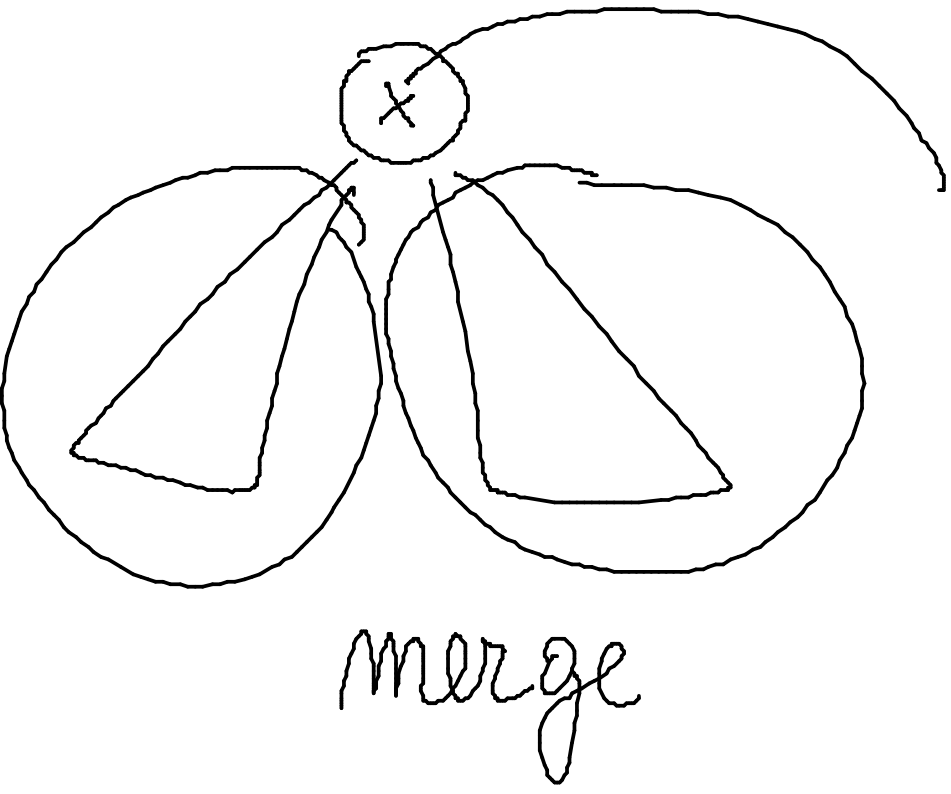
ins : 'a list → 'a → 'a list

ins l a inserisce a nella lista ordinata l
in modo che il risultato sia una lista
ancora ordinata (non decrescente)

Usare ins per definire linord : 'a btree → 'a list
che costruisce una lista ordinata dall'albero
binario

linord 

= [-3; -2; 1; 10; 20]


merge

```
let rec ins l a = match l with
      [] -> [a]
    | x::xs -> if a <= x then a::x::xs else x:: ins xs a ;;

let rec merge l1 l2 = match l1 with
      [] -> l2
    | x::xs -> ins (merge xs l2) x ;;

let rec fun bt = match bt with
    Void -> []
  | Node(x, lbt, rbt) -> ins (merge (fun lbt) (fun rbt)) x ;;
```