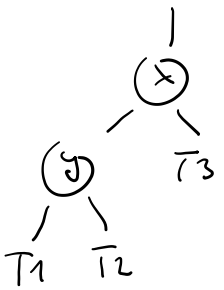
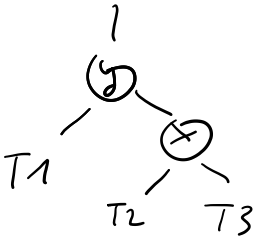
Zad 1.

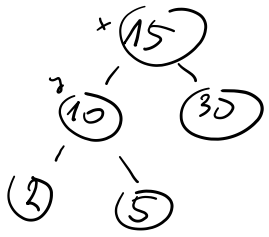
RIGHT-ROTATE(T,x)

1. y = x.left
2. x.left = y.right
3. if y.right ≠ NIL
4. y.right.p = x
5. if x.p == NIL
6. T.root = y
7. else if x == x.p.right
8. x.p.right = y
9. else x.p.left = y
10. y.right = x
11. y.p = x.p
12. x.p = y

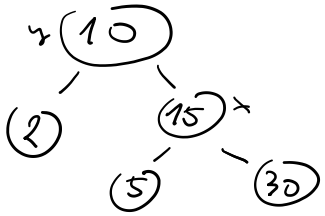


RIGHT-ROTATE(T,x) →





→



Zad 2.

```
AVL_insert(root, key)
    if not root
        return treeNode(key)
    else if key < root.value
        root.left = insert(root.left, key)
    else
        root.right = insert(root.right, key)

root.height = 1 + max(getHeight(root.left), getHeight(root.right))

balance = getBalance(root)

if (balance > 1 && key < root.left.value)
    return RIGHT_ROTATE(root)

if (balance < -1 && key > root.right.value)
    return LEFT-ROTATE(root)

if (balance > 1 && key > root.left.value)
    root.left = LEFT-ROTATE(root.left)
    return RIGHT-ROTATE(root)

if (balance < -1 && key < root.right.value)
    root.right = RIGHT-ROTATE(root.right)
    return LEFT-ROTATE(root)

return root
```

$\Rightarrow$ VSA je $O(n \log n)$

$O(n)$ → zbog inserta

$O(\log n)$ → zbog visine

$\Rightarrow O(n \log n)$