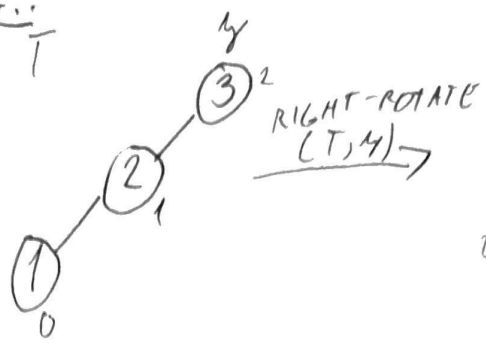
DZ 5

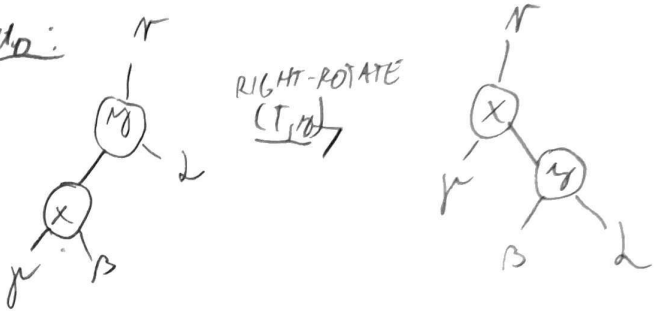
1.

```
RIGHT-ROTATE(T, y)
1.  x = y.left
2.  y.left = x.right
3.  if x.right ≠ NIL
4.      x.right.p = y
5.  if y.p == NIL
6.      T.root = x
7.  else if y == y.p.right
8.      y.p.right = x
9.  else y.p.left = x
10. x.right = y; x.p = y.p
11. y.p = x
```

Pr.:



ogledalo:



2.

```
INSERT(T, x, key)
1.  if x == NIL
2.      x.h = key
3.      x.h = 1
4.      x.left = x.right = x.p = NIL
5.  else if key < x.k
6.      x.left = INSERT(T, x.left, key)
7.      x.left.p = x
8.  else if key > x.k
9.      x.right = INSERT(T, x.right, key)
10.     x.right.p = x
11. else return x // ne dozvoljavamo duplikate
12. x.h = 1 + max(x.left.h, x.right.h)
13. balance = BALANCEOFNODE(x)
14. if balance > 1 and key < x.left.k // left left
15.     return RIGHT-ROTATE(T, x)
16. if balance < -1 and key > x.right.k // right right
17.     return LEFT-ROTATE(T, x)
    // left right
18. if balance > 1 and key > x.left.k
19.     x.left = LEFT-ROTATE(T, x.left)
20.     return RIGHT-ROTATE(T, x)
    // right left
21. if balance < -1 and key < x.right.k
22.     x.right = RIGHT-ROTATE(T, x.right)
23.     return LEFT-ROTATE(T, x)
24. return x
```

ANALIZA:

Na satu smo analizirali da su RIGHT-ROTATE i LEFT-ROTATE $O(1)$. Kod inserta prvih 11 linija je obično ubrzanje u BST što je kod balansiranog stabla poput AVL-a $O(\lg n)$, jer će balansirano stablo uvijek ostati $O(\lg n)$. Provjeravanje faktora balansiranosti i update nad vršnom čvora je $O(1)$. Do kraja je rađeno samo LEFT ili RIGHT-ROTATE koji su $O(1)$, pa je ukupna složenost algoritma $O(\lg n)$.