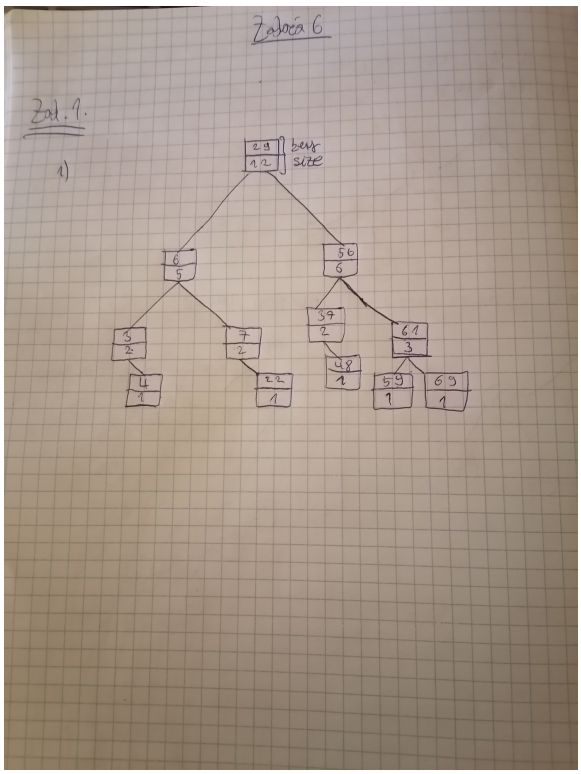
Zadaća 1

Zadatak 1

1)



2)

In [ ]:
```python
# pseudokod os-select procedure
def os_select(x, i):
    r = x.left.size + 1
    if(r == i):
        return x
    if(i < r):
        return os_select(x.left, i)
    else:
        return os_select(x.right, i - r)

#pojasnjenje na trazenom primjeru: os_select(root, 8)
# 1. x = [29]. r = x.left.size = 6. i > r => os_select(x.right, i = i-r = 2)
# 2. x = [56]. r = x.left.size + 1 = 3. i < r => os_select(x.left, i = 2)
# 3. x = [37]. r = x.left.size + 1 = 1; i > r => os_select(x.right, i = i -
# 3. x = [48]. r = x.left.size + 1 = 1; i == r => return x;
```

3)

In [ ]:
```python
# pseudokod os-rank procedure
def os_rank(T, x):
    r = x.left.size + 1
    y = x
    while y != T.root:
        #ako je y desno dijete svog roditelja
        #rangu dodaj velicinu lijevog podstabla + 1
        if y == y.p.right:
            r += y.p.left.size + 1
        y = y.p
    return r

#pojasnjenje na trazenom primjeru: os_rank(T, x=[59])
# 1. x = [59]. r = x.left.size + 1 = 1; y = [59];
# 2. y == y.p.left => r = 1. y = y.p = [61];
# 3. y == y.p.right => r = r + 1 + y.p.left.size = 4; y = y.p = [56];
# 4. y == y.p.right => r = r + 1 + y.p.left.size = 5 + 5 = 10; y = y.p = [29
# 5. y == T.root => return r = 10;
```

Zadatak 2

In [ ]:
```python
#neka vrsta aliasa
null = None

def os_select_iter(x, i):
    #tu se stitim od podvaljenog nullptr-a
    if(x == null):
        return x
    y = x #postavi pomocni pointer y na x
    j = i #prekopiraj i u novi brojac j

    #rank proslijedenog cvora. Ukoliko odmah odgovara nece niti uci u petlju
    r = y.left.size + 1

    #dok nisi dosao na pravi rankg ili dosao do nullptr-a
    while(j != r and y != null):
        #relativan rank trenutnog cvora y
        r = y.left.size + 1

        #j < r => pravi cvor se mora nalaziti negdje lijevo
        if(j < r):
            y = y.left
        #j > r => pravi cvor se mora nalaziti negdje desno
        #trazeni rank je sad j - r
        if(j > r):
            y = y.right
            j = j - r

    #bez obzira je li null ili ne, vrati y
    return y
```

Zadatak 3

In [ ]:
```
# pretpostavimo da search funkcija AVL-a vraca null
# ako nije nasla cvor unutar stabla
def os_key_rank(T, k):
    x = T.search(k)

    # niti jedan cvor ne moze imati rank -1
    # pa odmah znamo da nesto ne stima
    if(x == null):
        return -1
    return os_rank(T, x)

#VRIJEME IZVRSAVANJA: O(lg n), jer search je O(lg n) + rank je O (lg n)
```

Zadatak 4

1)

In [ ]:
```
#ith successor koristeci nove operacije na augmentiranom avl-u
def ithSuccessor(T, x, i):
    #dohvati rank proslijedenog nodea
    j = os_rank(T, x)

    #ako je i = 0 to je kao da trazis rank
    #od x, vrati x
    if(i == 0):
        return x

    #inace, njegov i-ti sljedbenik ima rank j + i
    return os_select(T.root, j + i)

#VRIJEME IZVRSAVANJA: O(lg n) za os-rank + O(lgn) za os-select = O(lg n)
```

2)

In [ ]:
```
#ith successor koristeci stare operacije na augmentiranom AVL-u
def ithSuccessor(T, x, i):
    y = x
    for k in range(1, i + 1):
        y = tree_successor(T, x)
    return y
# VRIJEME IZVRSAVANJA
# ovaj algoritam koristi k uzastopnih poziva tree_successor funkcije
# za takav algoritam smo u jednoj od prethodnih zadaca pokazali da je njegov
# vrijeme izvrsavanja O(k + h), gdje je k broj prijeđenih grana.
# kako je visina stabla h, vrijeme izvrsavanja algoritma je O(k + lg n)
```