Question 1 (Exercise 13.2-1, Cormen et al.)

Write pseudo code for RIGHT-ROTATE.

```
RIGHT-ROTATE(T, y)
    x = y.left          // set x
    y.left = x.right    // turn y's left subtree into x's right subtree
    IF x.right ≠ T.nil
        x.right.p = y
    x.p = y.p           // Link y's parent to x
    IF y.p == T.nil
        T.root = x
    ELSEIF y == y.p.right
        y.p.right = x
    ELSE y.p.left = x
    x.right = y         // Put y on x's right
    y.p = x
```

Question 2 (Exercise 13.2-2, Cormen et al.)

Argue that in every n-node binary search tree, there are exactly (n − 1) possible rotations.

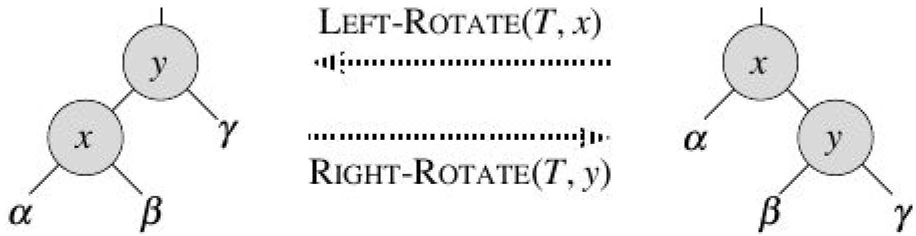
All nodes have a left and right child and therefore, at max there will always be 2 nodes that share a parent. Except for the root node. It doesn't share a parent with any other node.

Therefore, we can rotate all the other nodes using left and right rotates except this one node. (n - 1)

Question 3 (Exercise 13.2-3, Cormen et al.)

How do the depths of nodes in a BST change when a rotation is performed?

To analyze how the depths changes we can use the example from Cormen et al.



As we can see, when we `left rotate`, x decreases in depth while its right child y increases in depth as well as y's right child, gamma.

While if we `right rotate`, y decreases in depth while its left child, x, increases in depth as well as it's left child, alpha.

In both cases though, beta, doesn't change in depth. During `left rotation` it is the left child of y and during `right rotation` it is the right child of x.

Question 4 (Exercise 13.3-2, Cormen et al.)

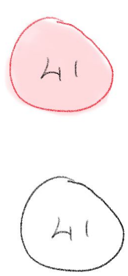
Write down or illustrate the red-black trees that result after successively inserting the keys 41; 38; 31; 12; 19; 8 into an initially empty red-black tree.

Cases:

1. Z = root -> color black
2. Z.uncle = red -> recolor grandparent, parent, and uncle
3. Z.uncle = black (triangle) -> rotate Z.parent
4. Z.uncle = black(line) -> rotate Z.grandparent and recolour grandparent, parent, and uncle

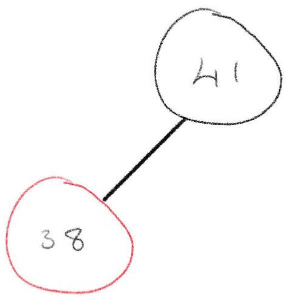
Inserting 41, 38, 31, 12, 19, 8

41

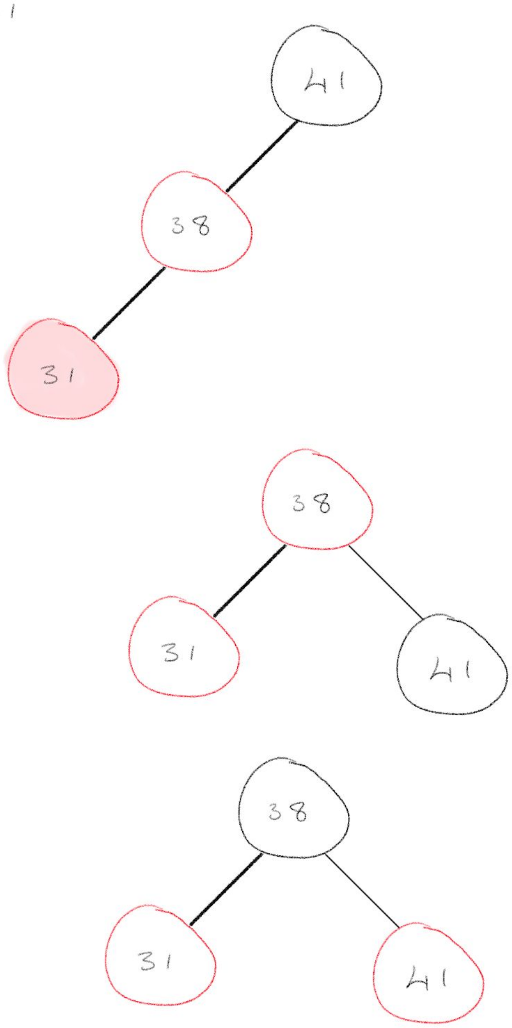


After inserting 41 case 0 occurs.

38

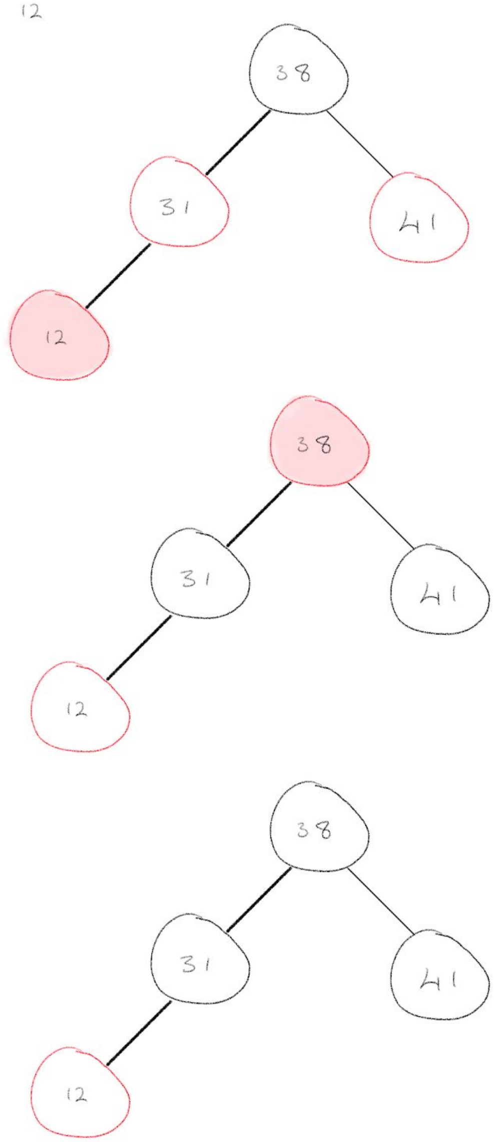


31



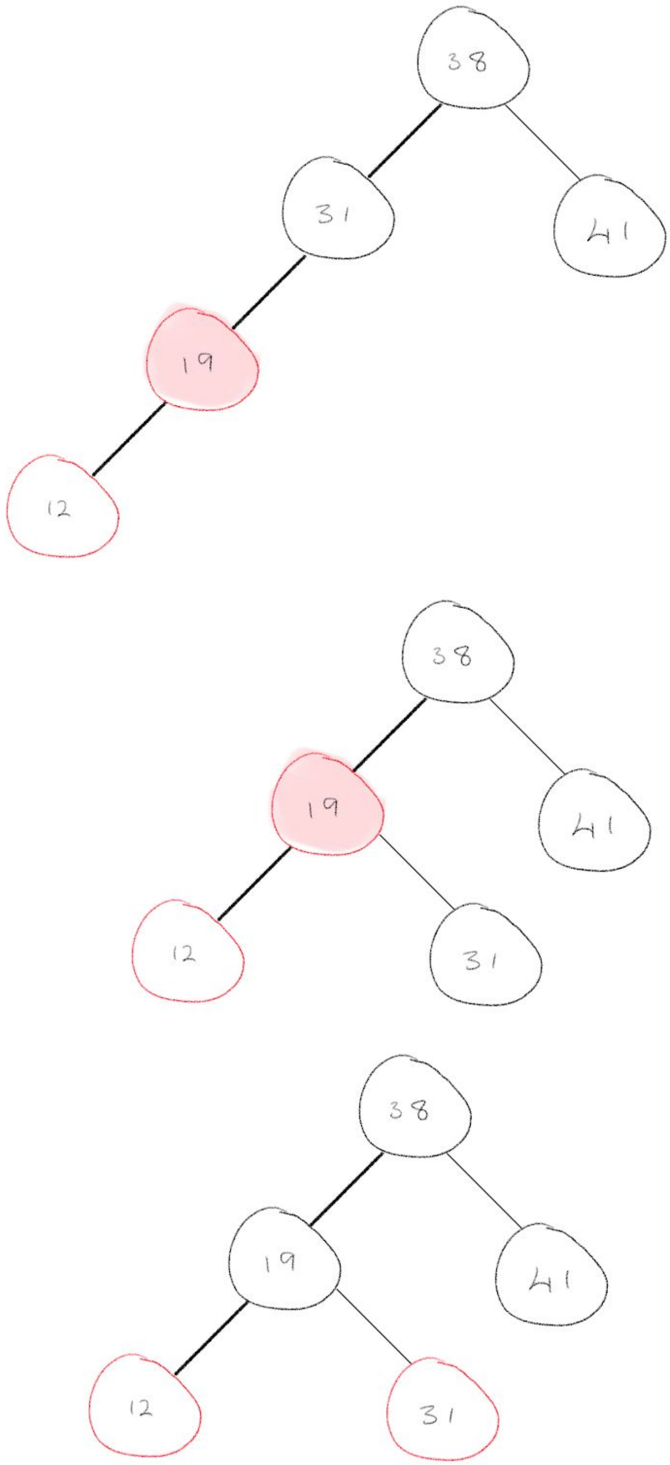
After inserting 31, case 3 occurs

12



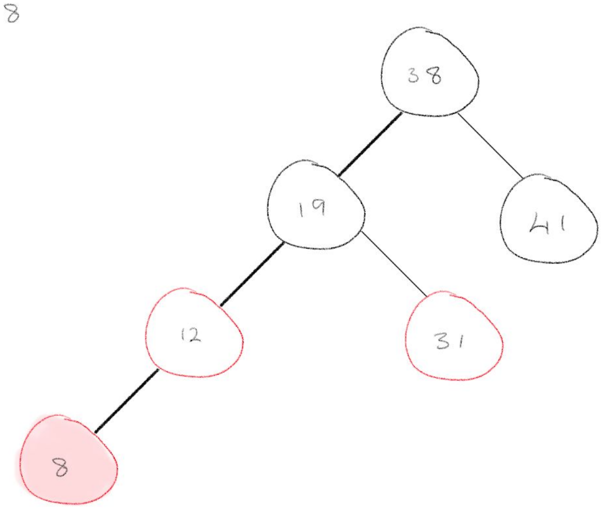
After inserting 12, case 1 occurs followed by case 0

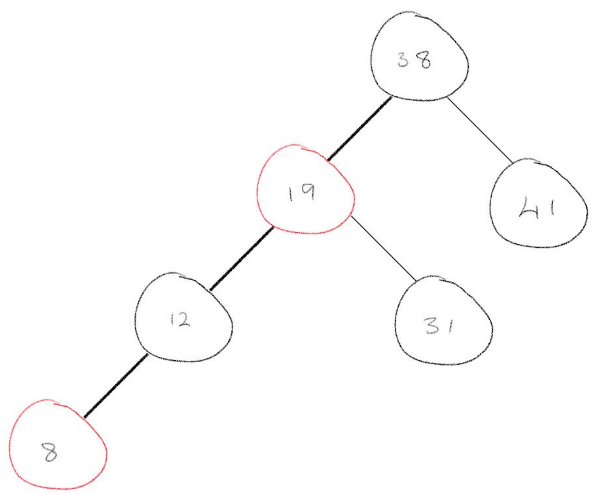
19



After inserting 19 case 2 occurs

8





After inserting 8, case 1 occurs