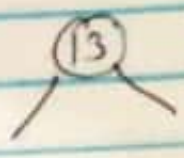
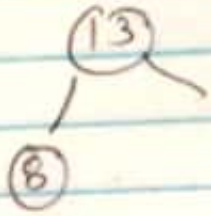
Q. 1.

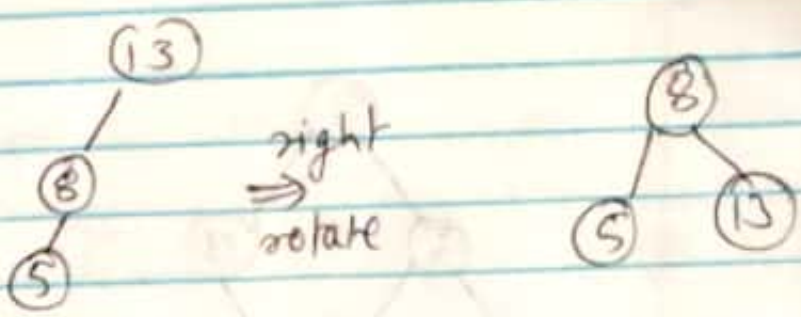
insert : 13



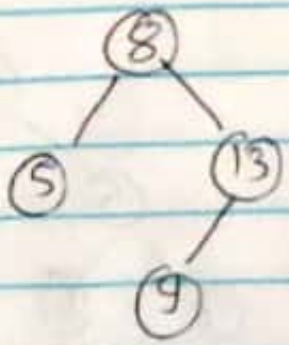
insert : 8



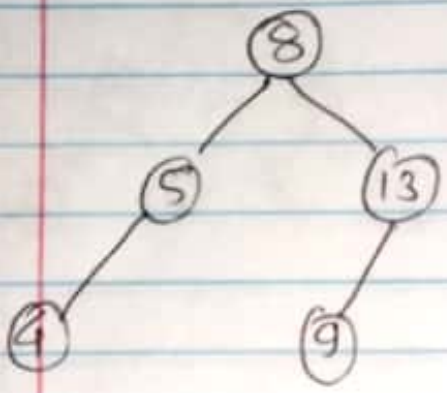
insert 5 :



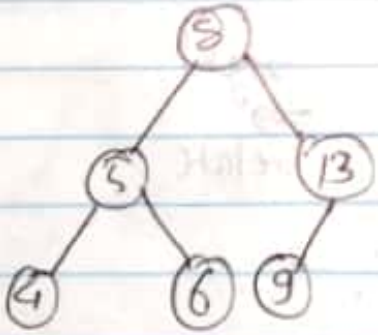
insert : 9



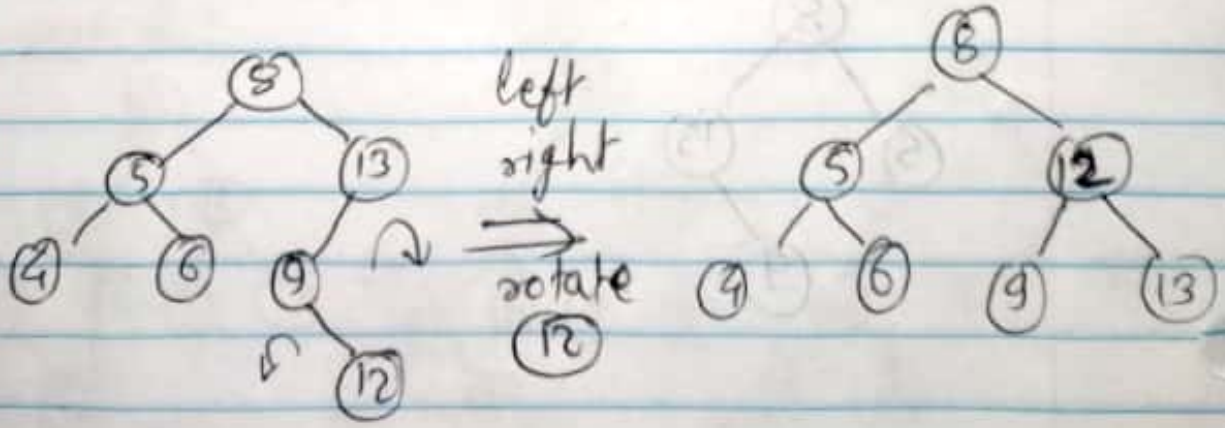
insert 4 :



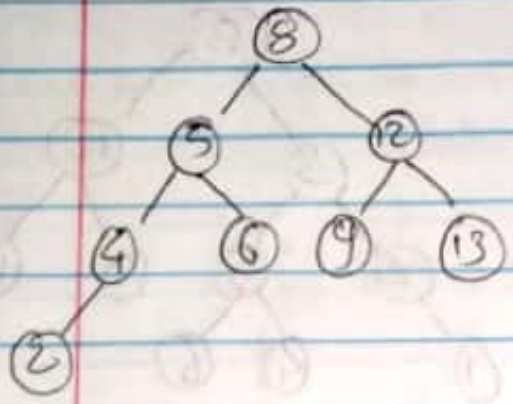
insert 6 :



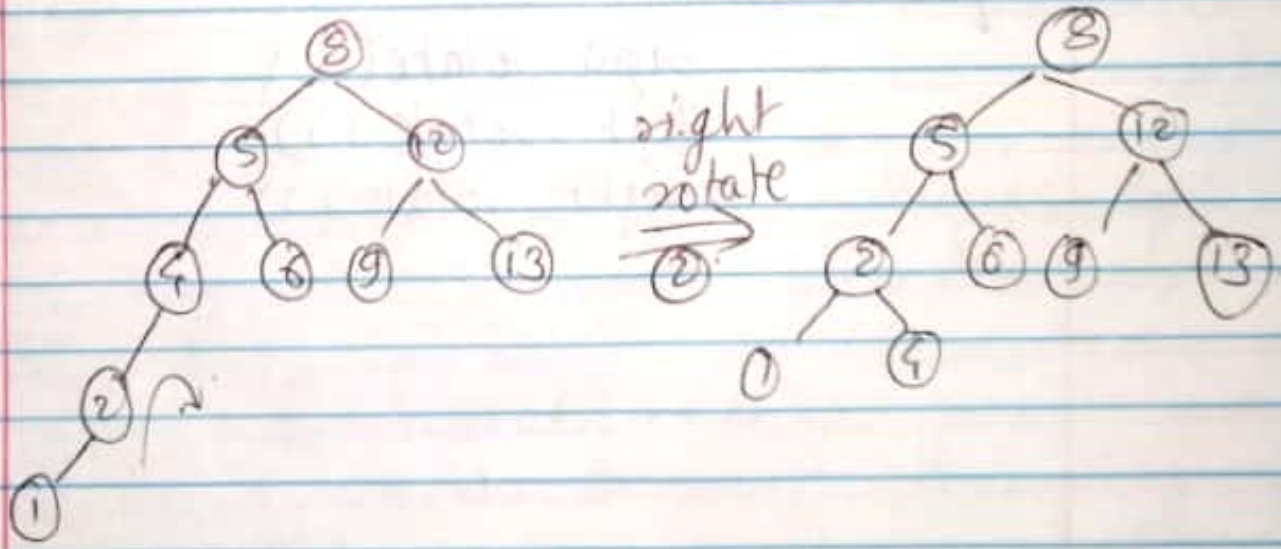
insert 12



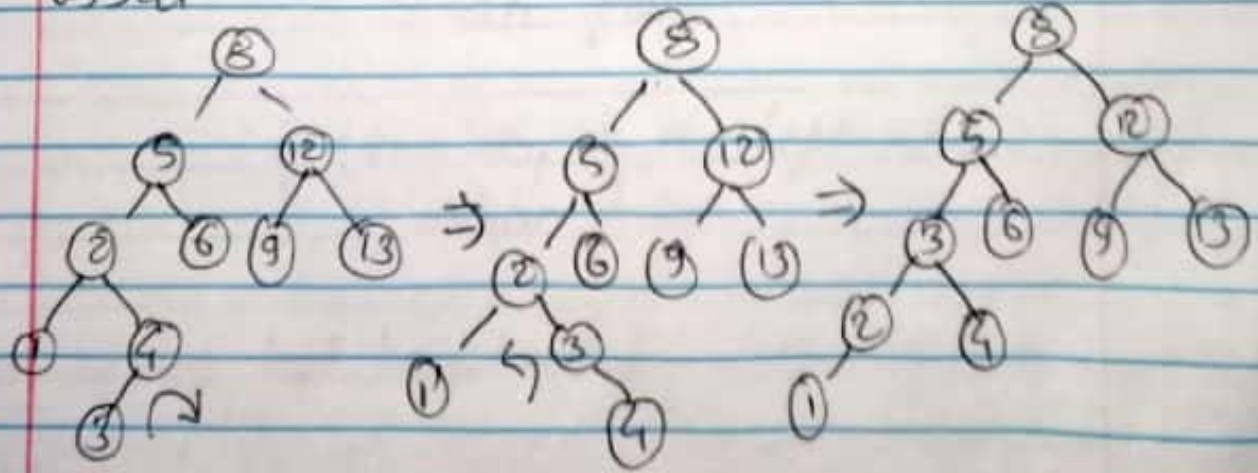
insert 2:



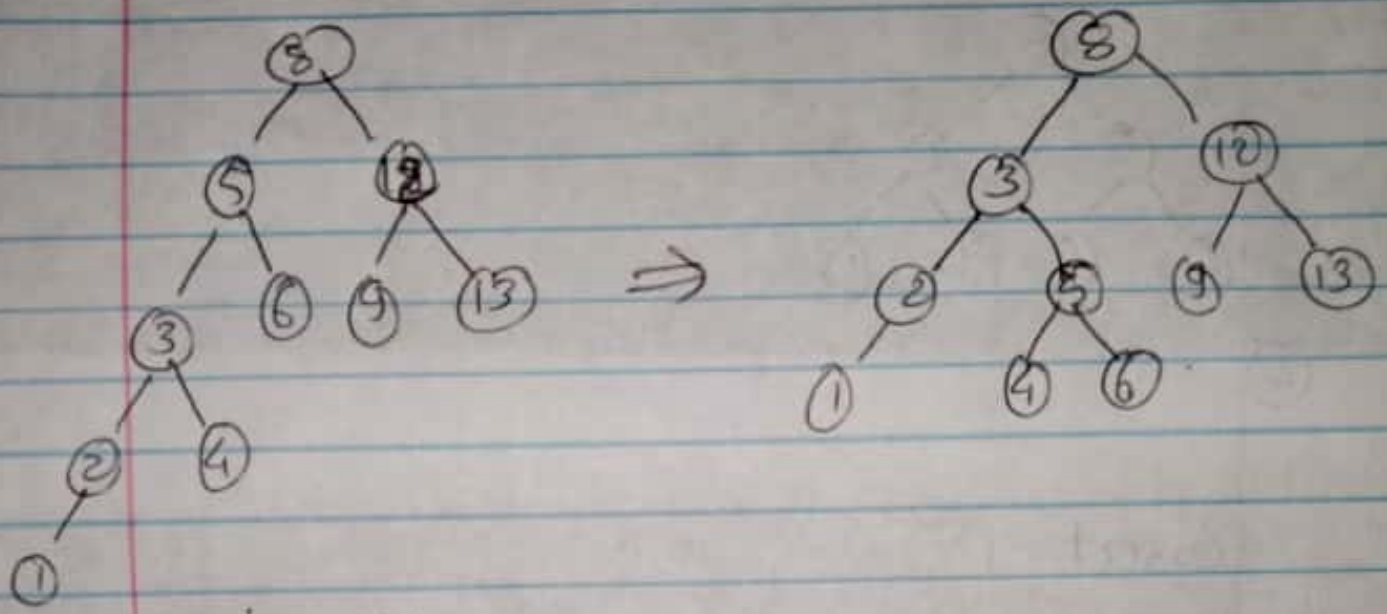
insert 1:



insert 3:



insert 3 continue:



for 3:

right - rotate(3)
left - rotate(3)
right - rotate(3).

Q.2.

```
bool is_AVL (Node n)
{
    if (n == 0)
        return true;

    if (n -> left == 0 && n -> right == null)
    {
        if n -> height == 0
            return true;
        else
            false
    }.

    int height_left = -1;      // checking left
    if (n -> left != 0)        // recursively
    {
        if (is_AVL (n -> left))
            return true
        else
            return false

        if (n -> key < n -> left -> key)
            return false

        left height_left = n -> left -> height
```

```
int right_height = -1
if (n->right != 0)
{
    if (is_AVL(n->right))
        return true
    else
        return false

    if (n->key > n->right->key)
        return false

    right_height = n->right->height
}

if (max(height_left, right_height)+1 !=
        n->height)
    return false

if (abs(height_left - right_height <= 1)
    return true
else
    false
```