Prem Atul Jethwa
10018618 10

Assignment ID: 02

Q1.)

a.) Total no. of nodes in full binary tree = $2^0 + 2^1 + 2^2 + \ldots 2^h$ (here h is the height of the Binary tree).

For height = 3

$n = (1+2+4)+8$ { $(1+2+4) = 7$ are the non-leaf node
8 are the number of leaf node
n is the total no. of nodes.

$n = 15$

For height = 2

$n = (1+2)+4$ { $(1+2) = 3$ are the no. of non leaf node
4 are no. of leaf node
n is the total no. of nodes.

$n = 7$

Hence, we observe that number of leaf nodes is one greater than number of non-leaf nodes in full binary tree.

greatest number of leaf nodes = floor $\left(\frac{n+1}{2}\right)$

.... floor division to get integer value.

If height of the tree is 3, we get in total 15 node

Therefore, $\left(\frac{15+1}{2}\right) = 8$ leaf nodes which is correct.

least number of leaf nodes = Root node can have a single left or right child, which is also interior node, so height will become 1 in this case, and there is exactly one child.

Least number of leaf nodes = 1.

b) Total no. of nodes in a full binary tree = no. of
II) non-leaf nodes + no. of leaf nodes.

No. of leaf nodes = n
No. of non-leaf nodes = n-1
Total no. of nodes = n + (n-1) = 2n-1

Therefore, no. of nodes in a full binary tree = 2n-1.
Where n is the no. of leaf nodes.

For ex- we know that if height of tree is 3, total no. of leaf nodes becomes 8.
Thus, number of nodes in a full binary tree
= 2 × 8 - 1 = 15.

Q2)

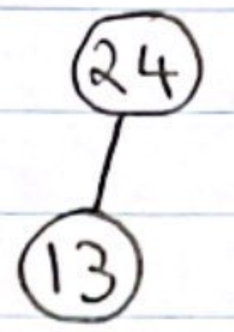
a) Algorithm for binary search tree:
I) Step1: insert an element and assign it to root if tree is empty.
Step2: If tree is not empty.
Case 1: if the new element is less then parent insert in left subtree.
Case 2: if the new element is less then parent insert in right subtree.
Step 3: Repeat till all elements in the array are inserted.

Below are steps for binary search tree.

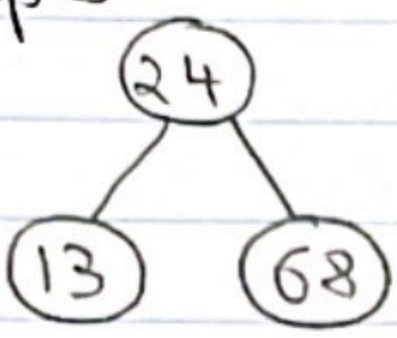
24, 13, 68, 62, 66, 22, 58, 80, 2, 59, 23, 54, 1, 15, 95

Step 1

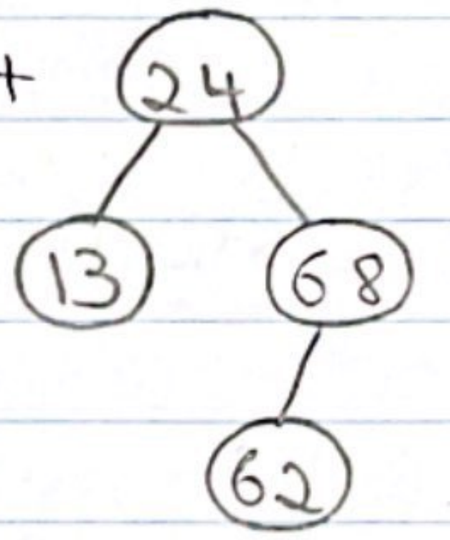
Step 2



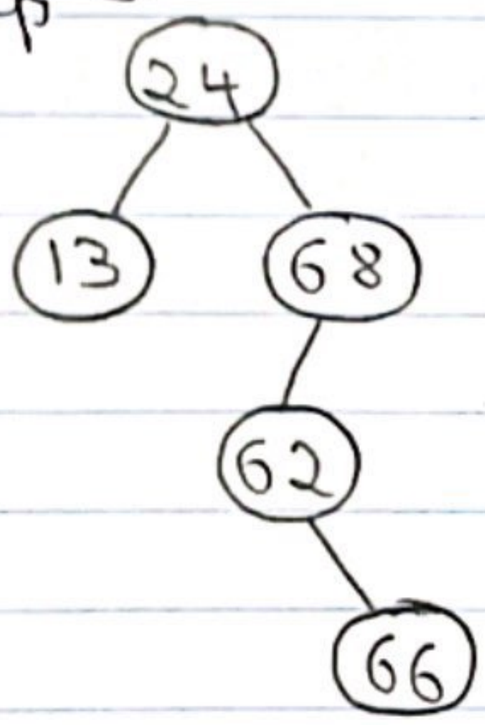
Step 3



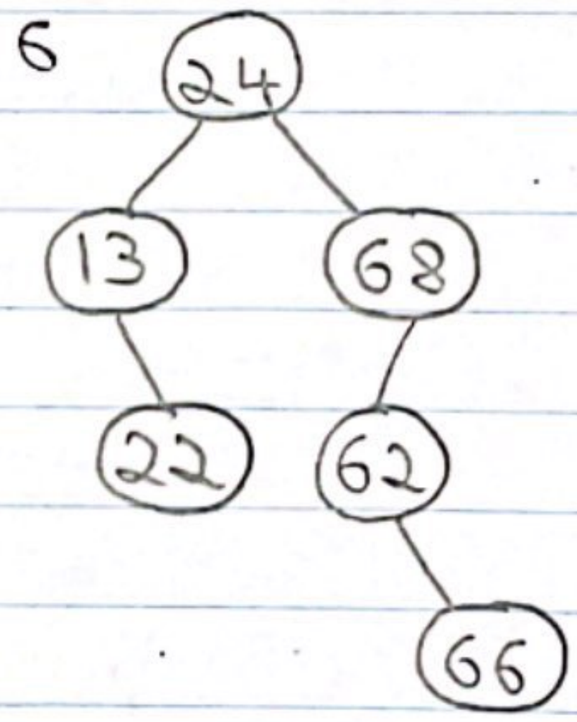
Step 4



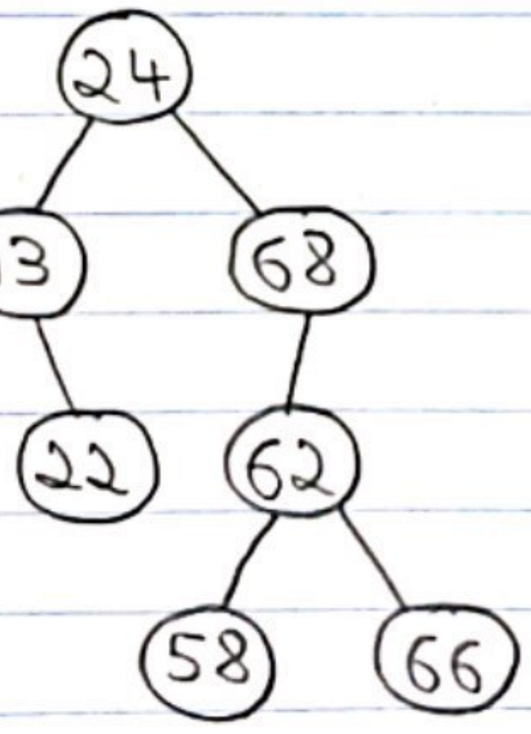
Step 5



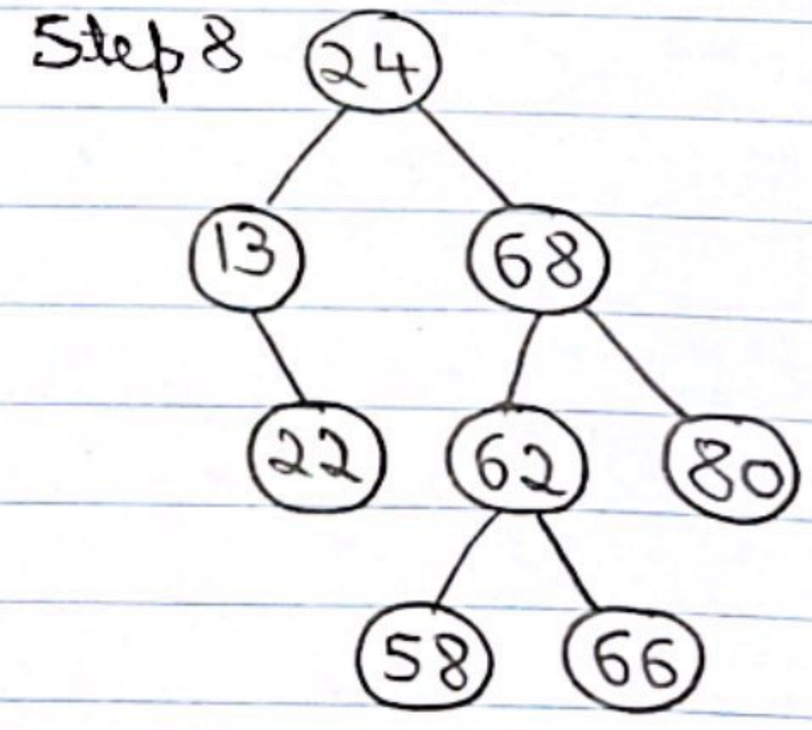
Step 6



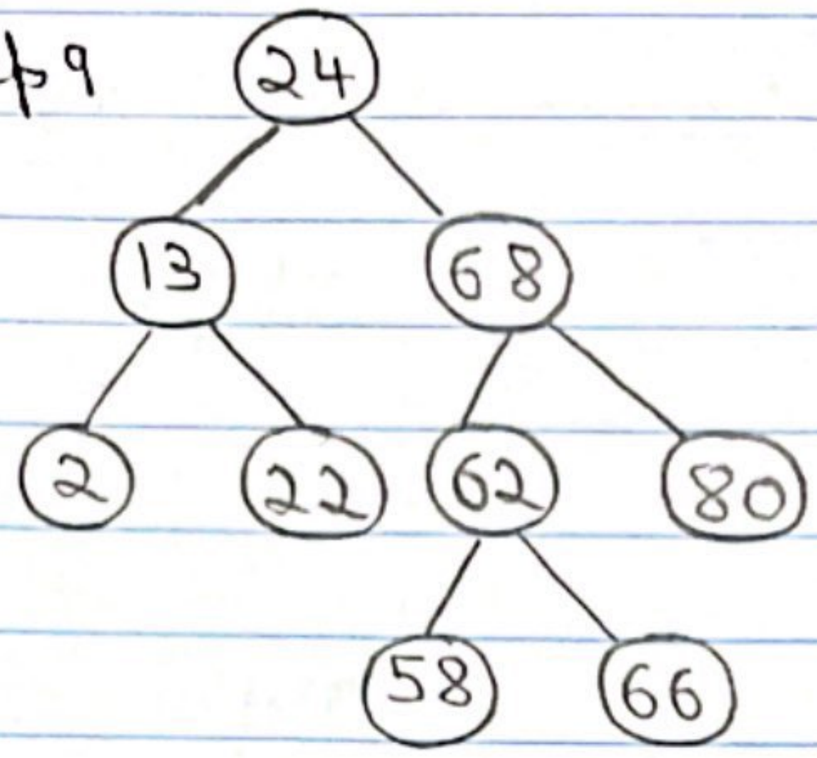
Step 7



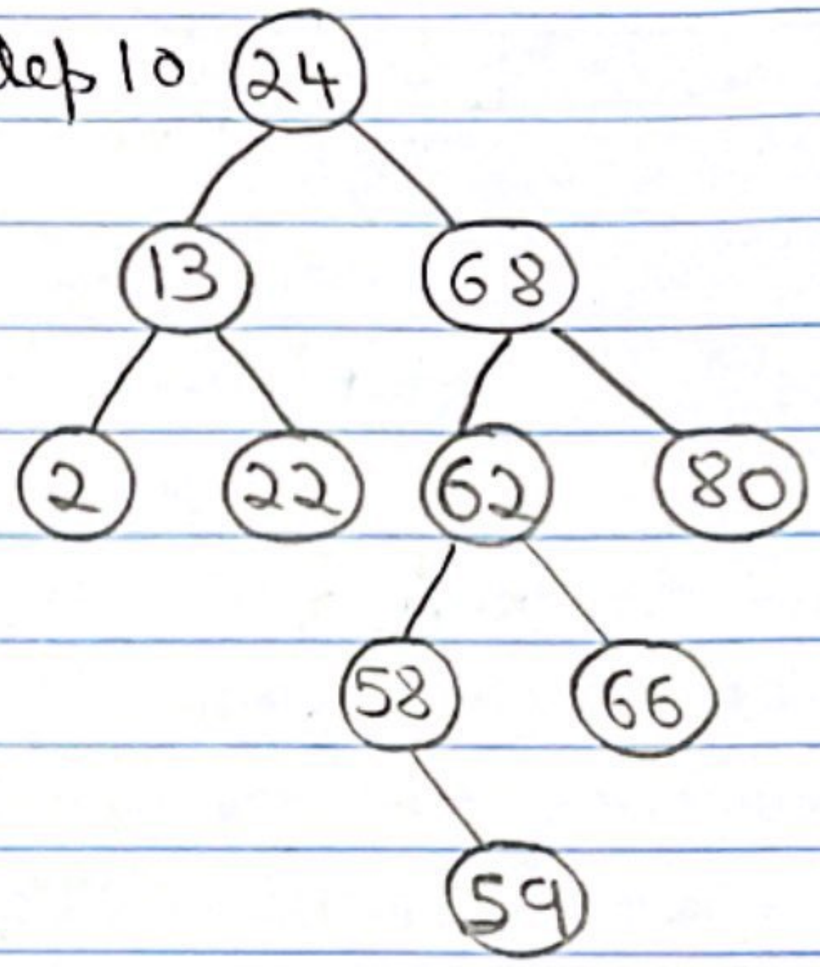
Step 8



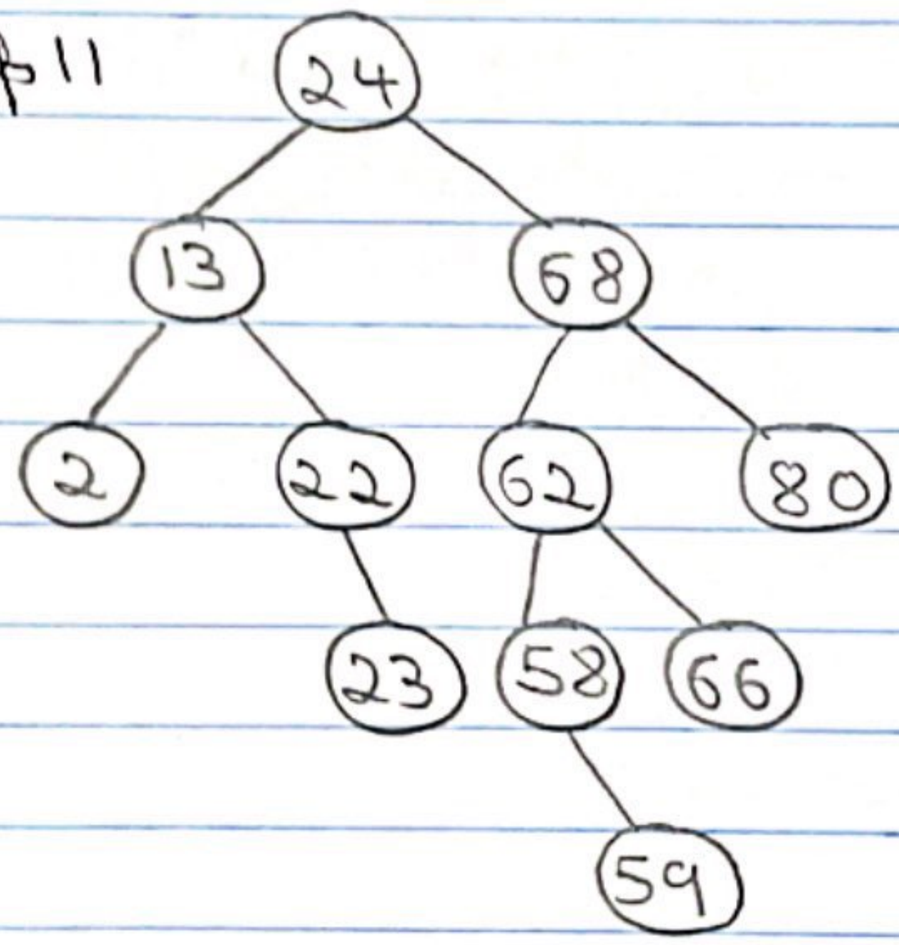
Step 9



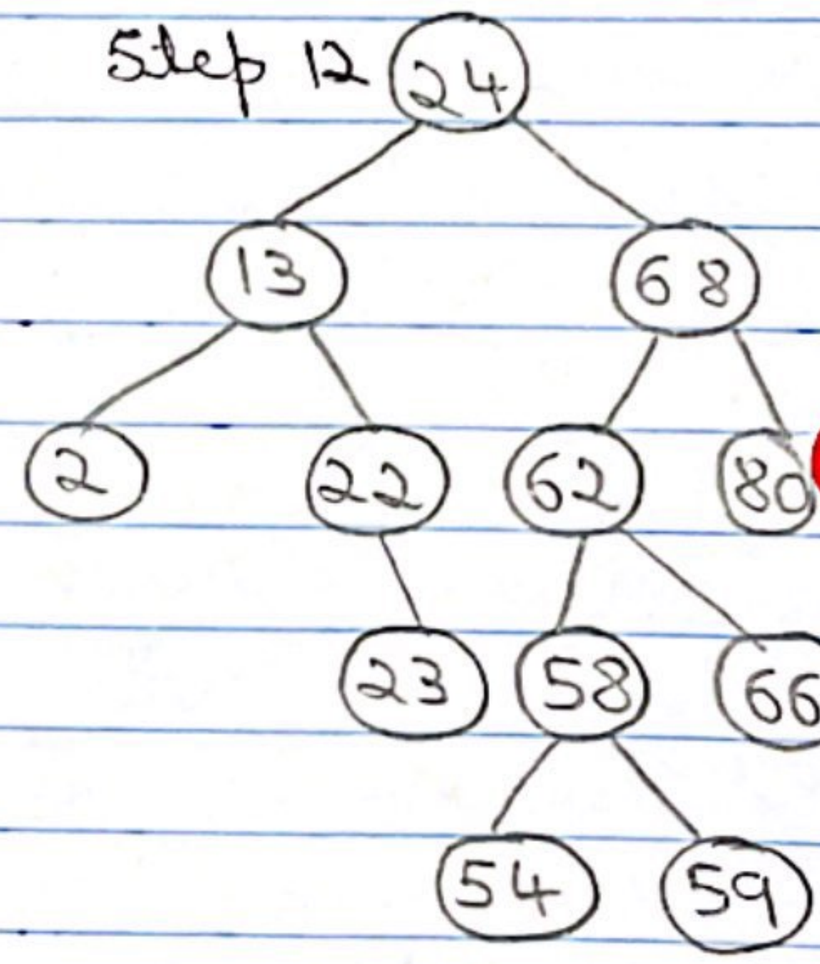
Step 10



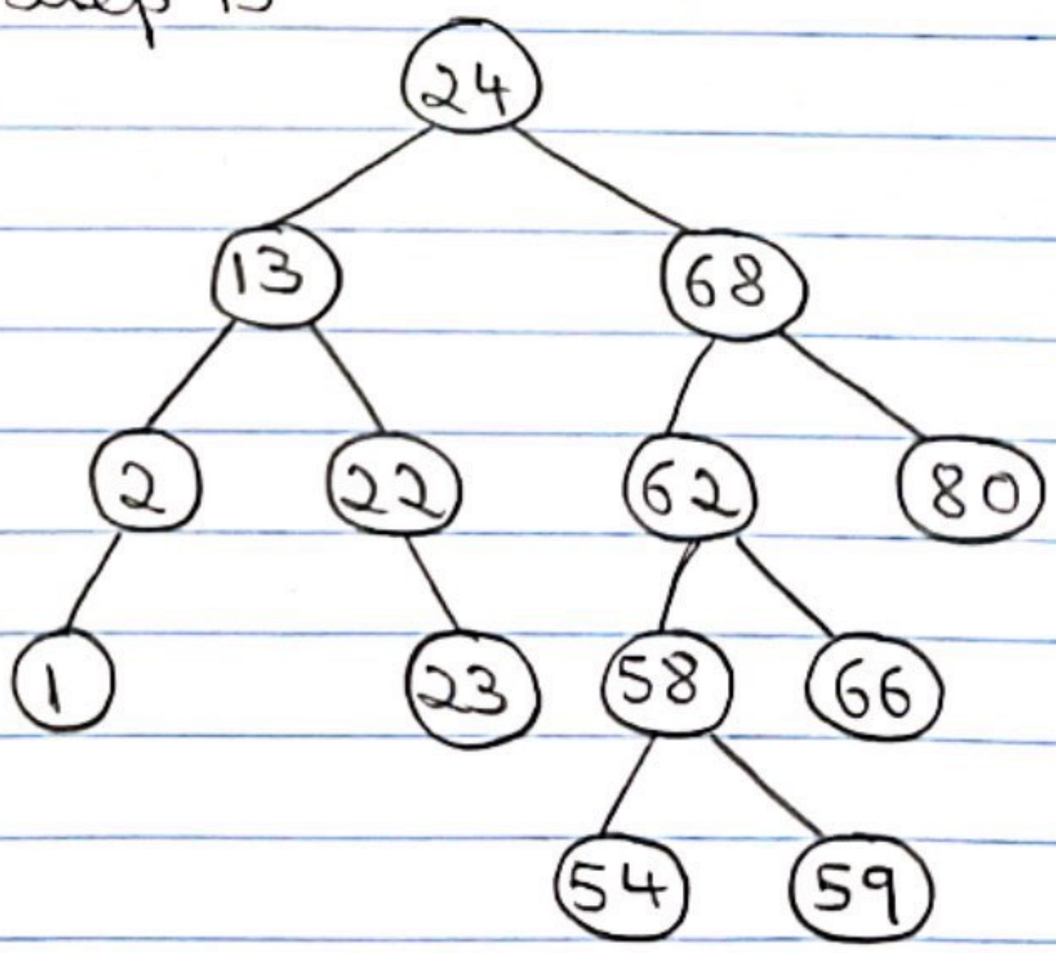
Step 11



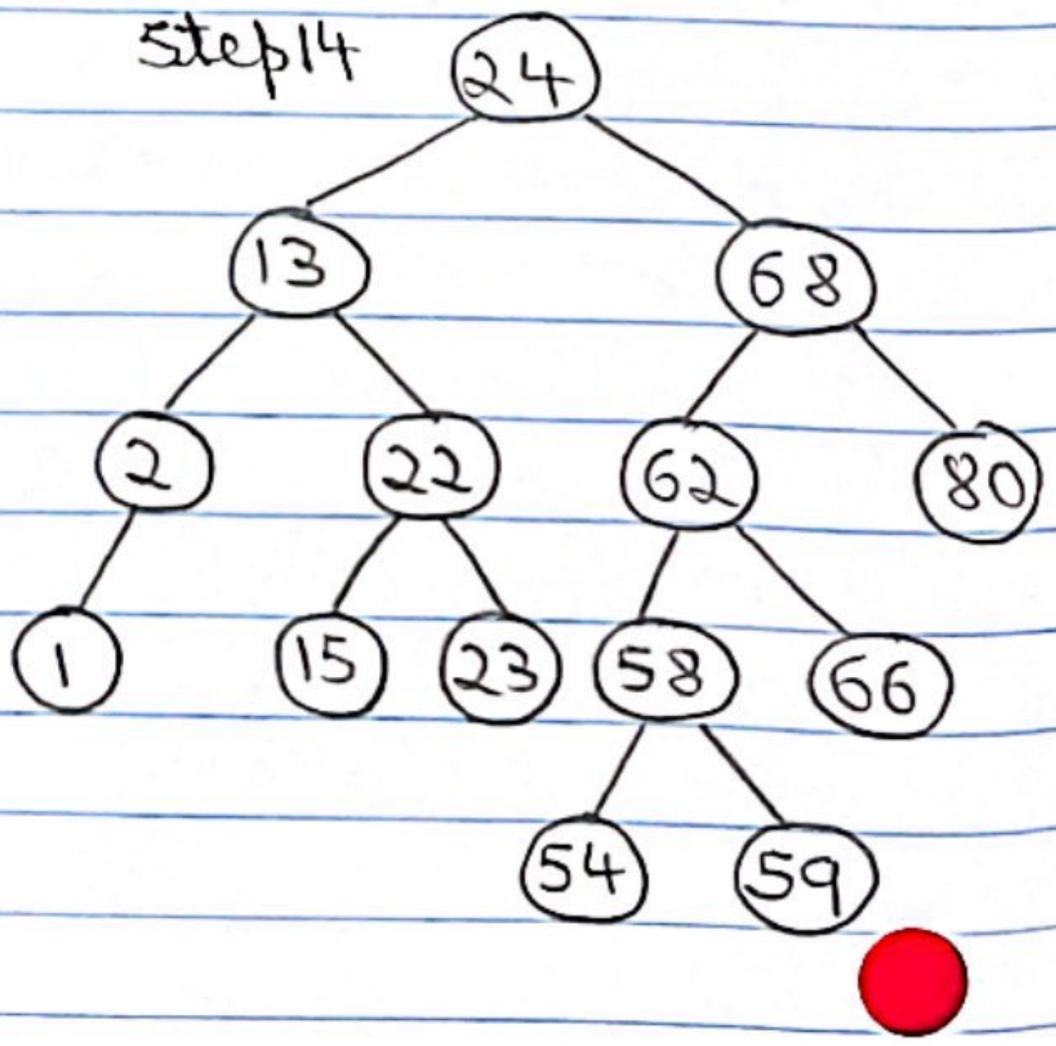
Step 12



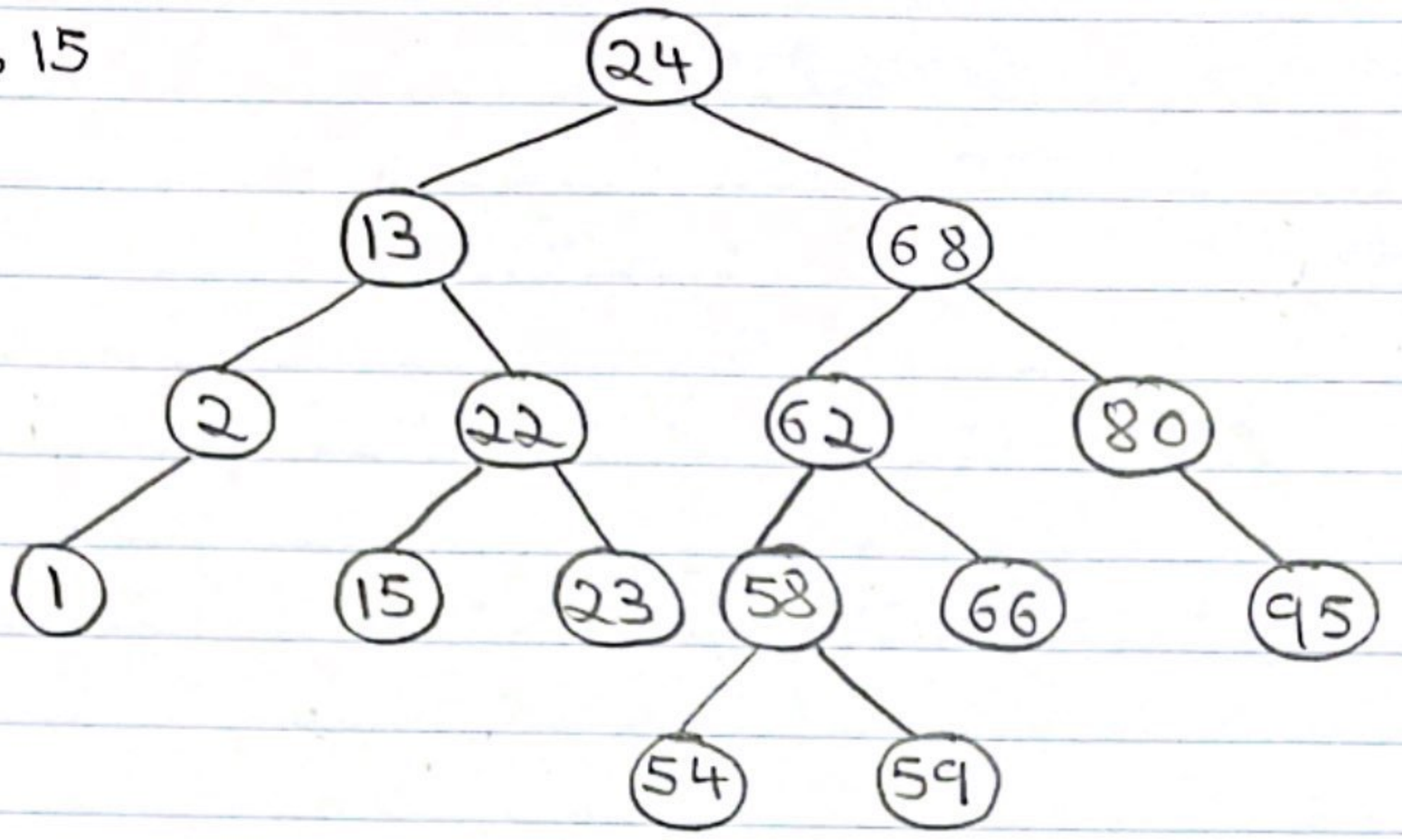
Step 13



Step 14



Step 15

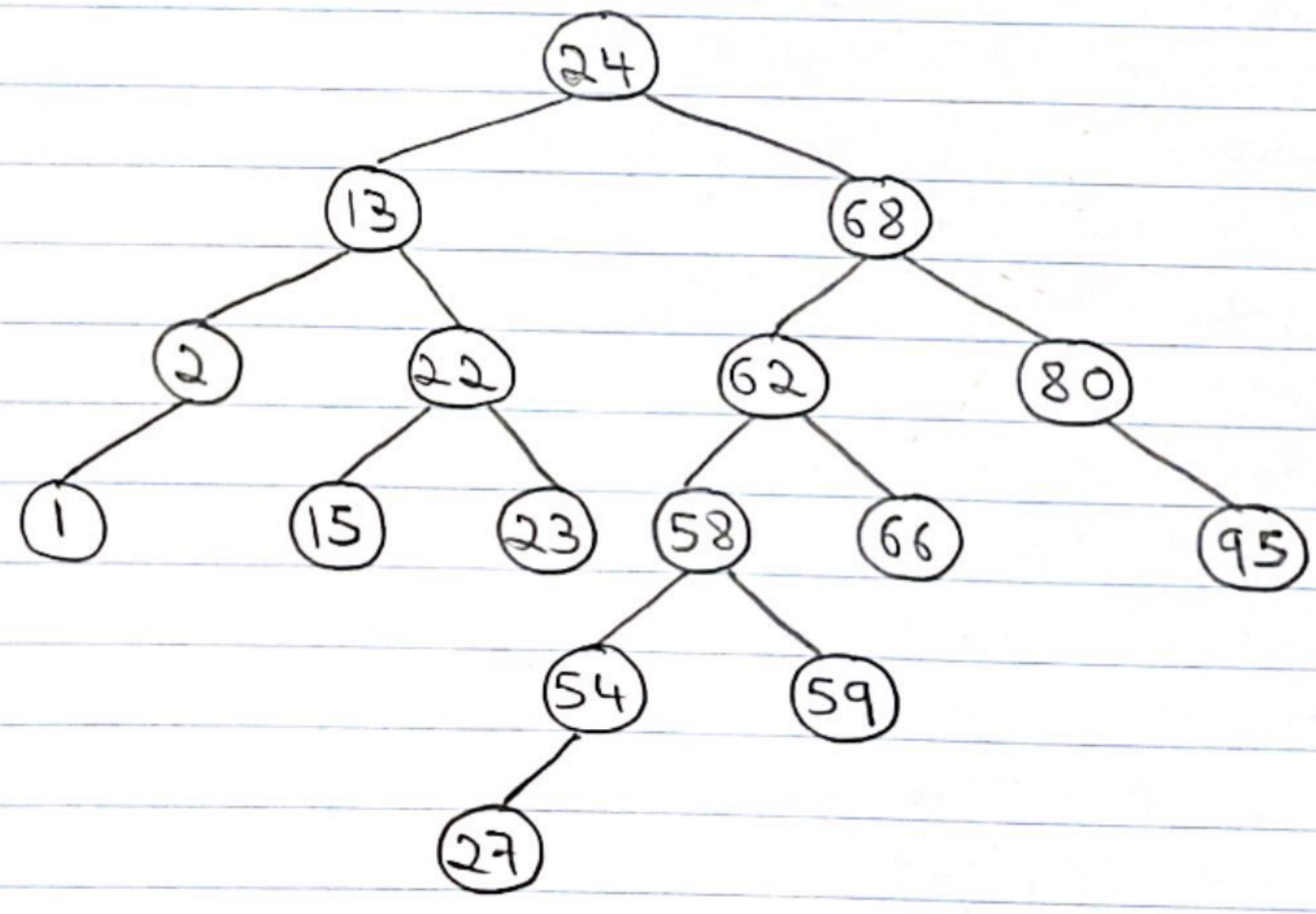


At <u>step 15</u> we obtain a <u>binary tree</u>.

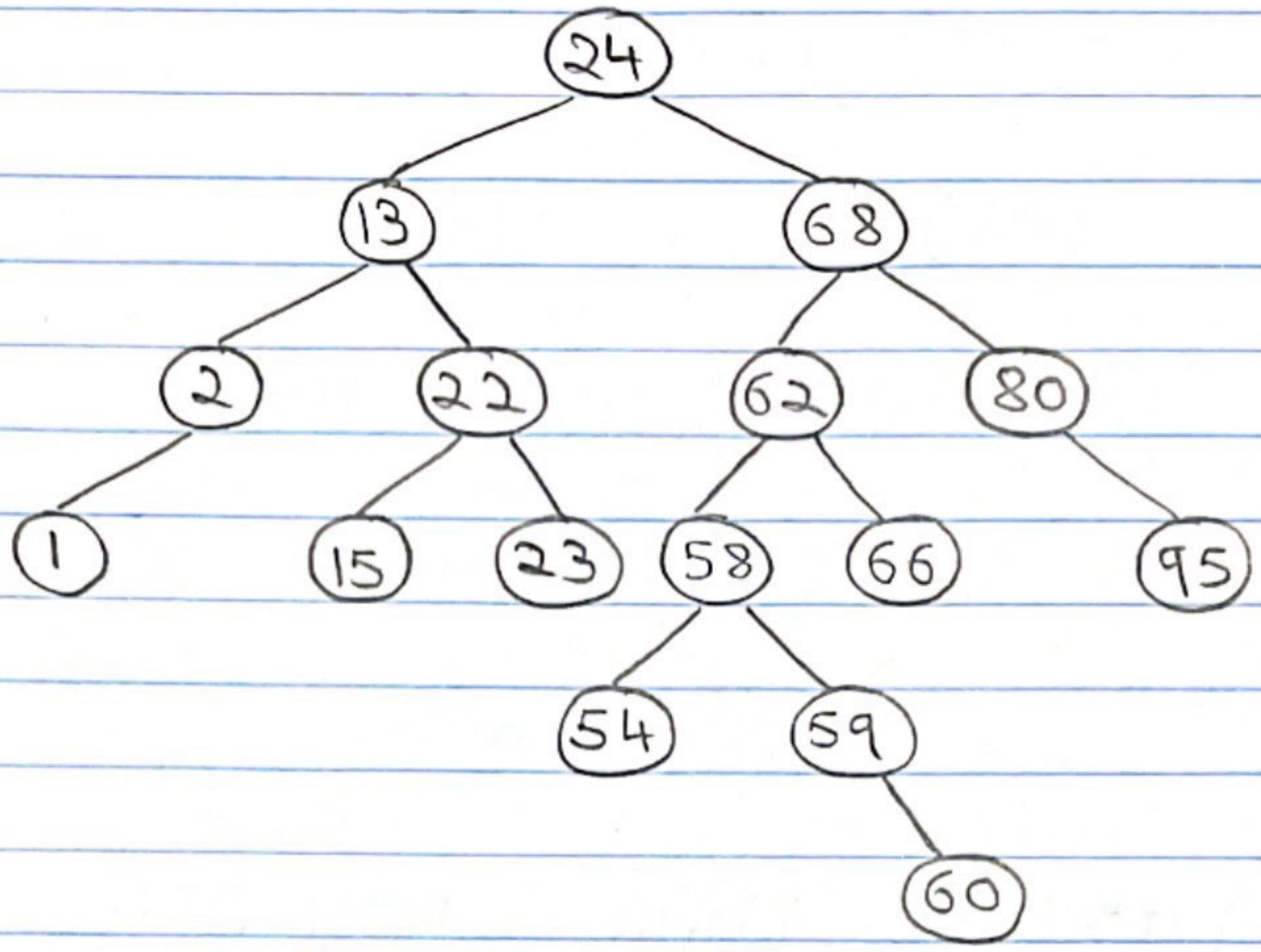
b.) II.) The integers that will increase the height of given tree are:

27 , 60 , 29 , 56 , 61

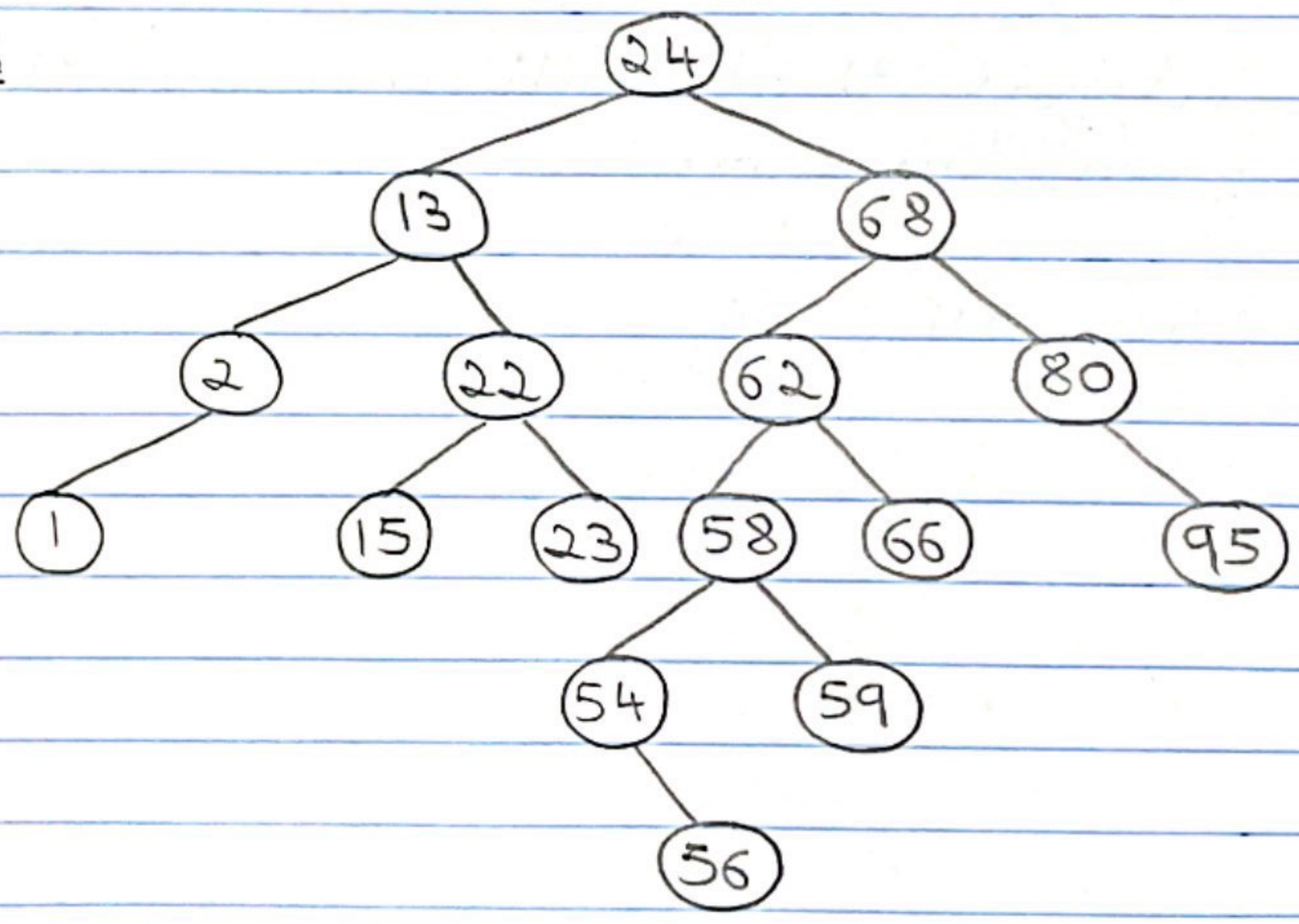
<u>27</u>



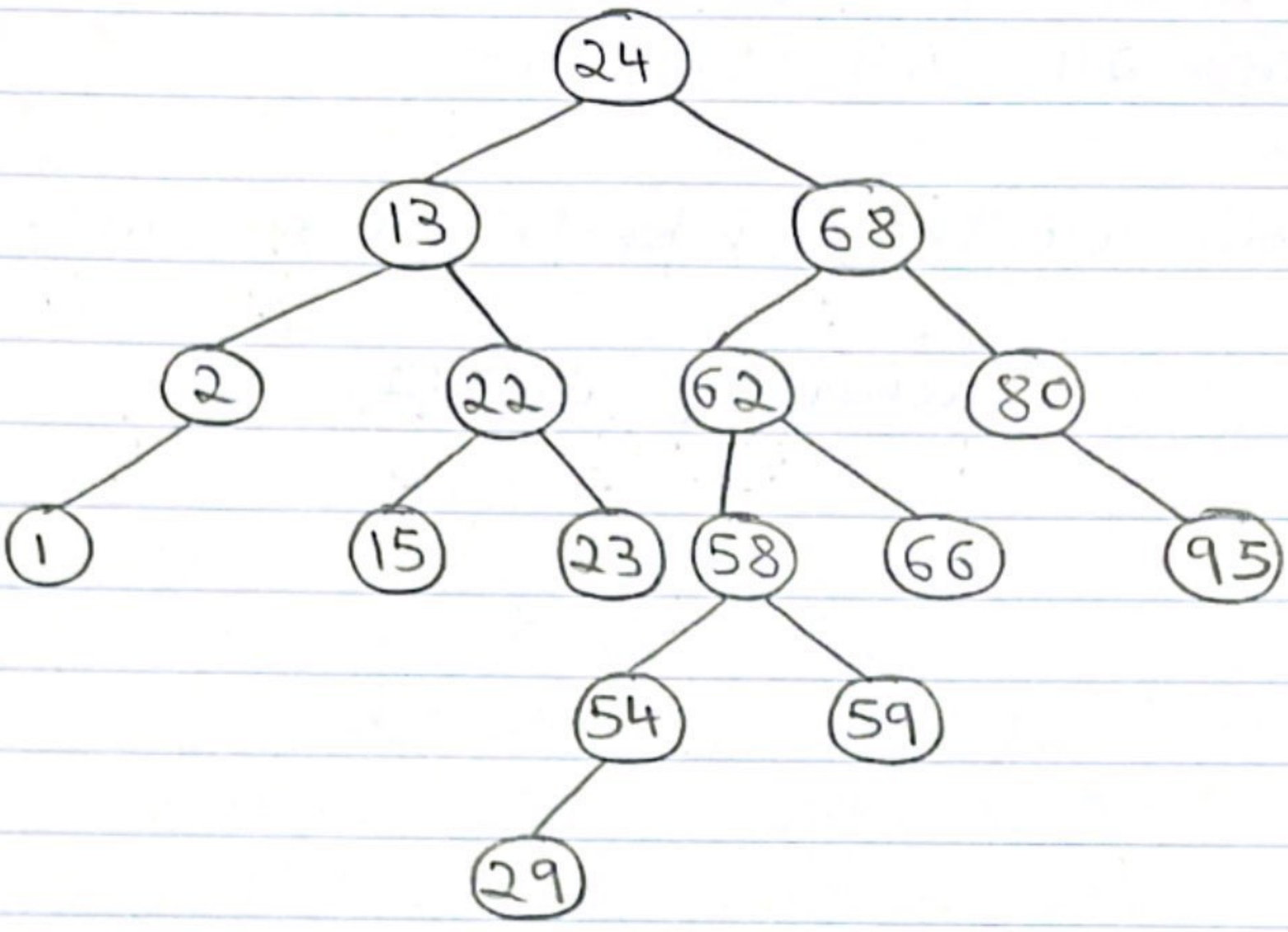
60



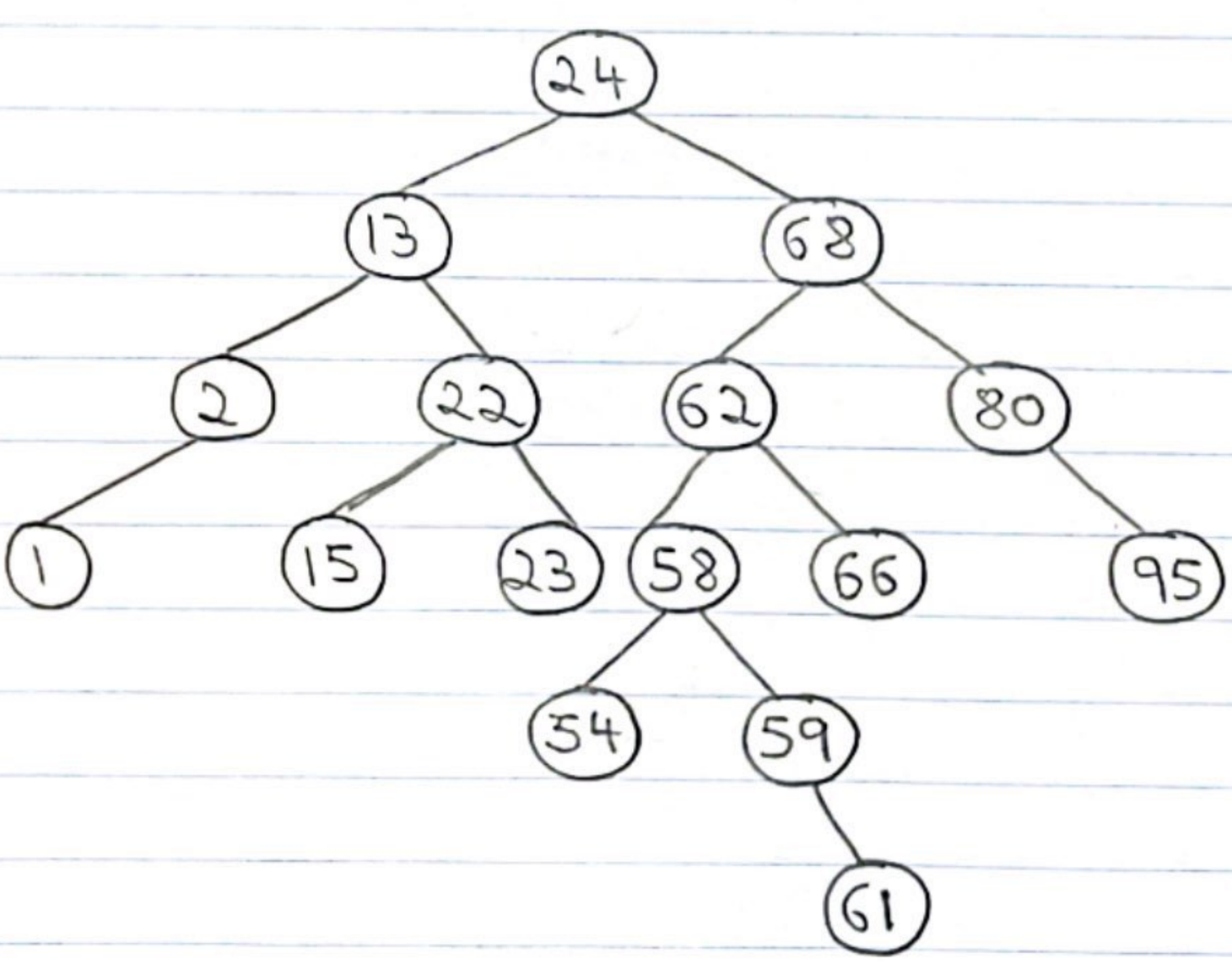
56



29



61

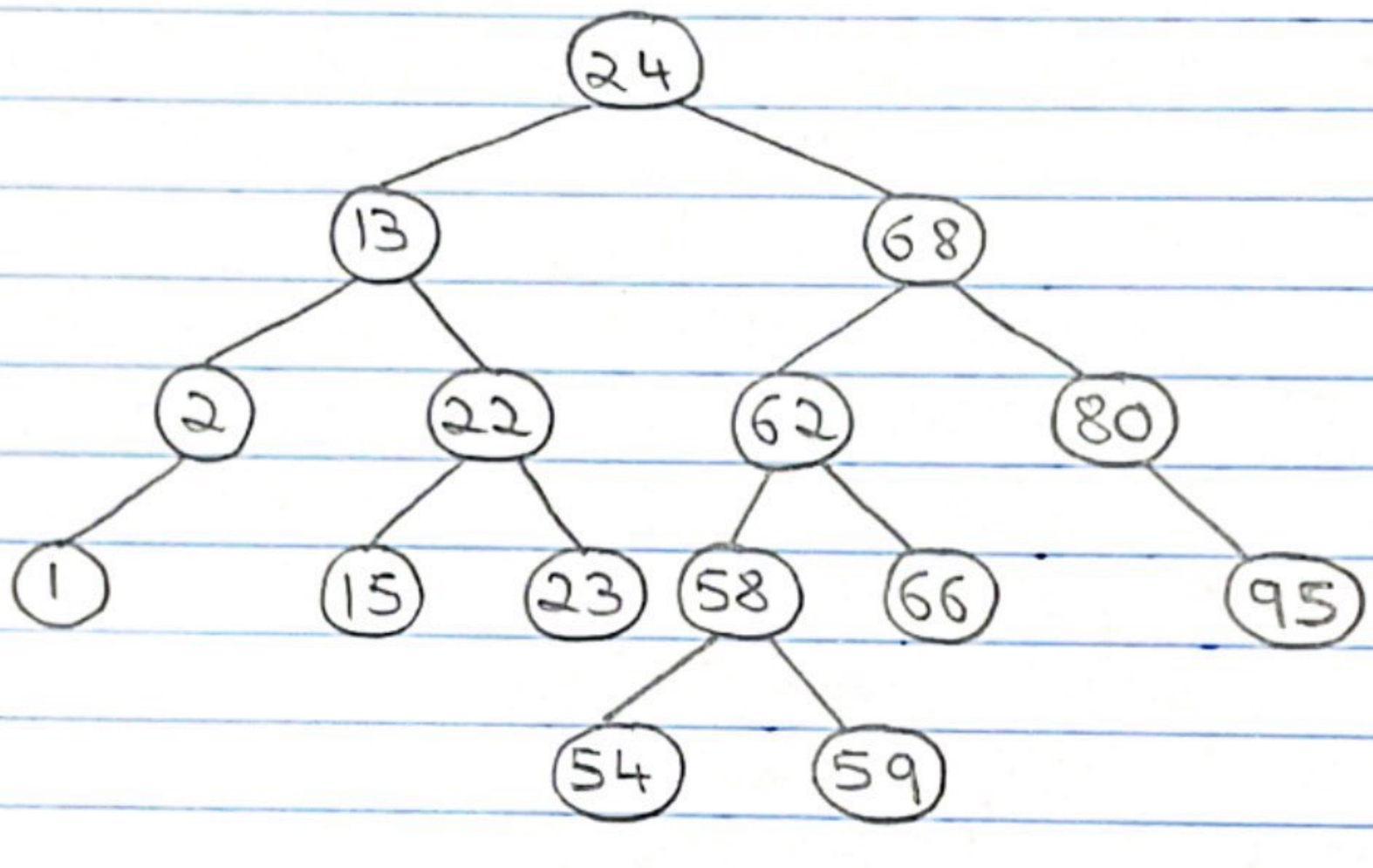


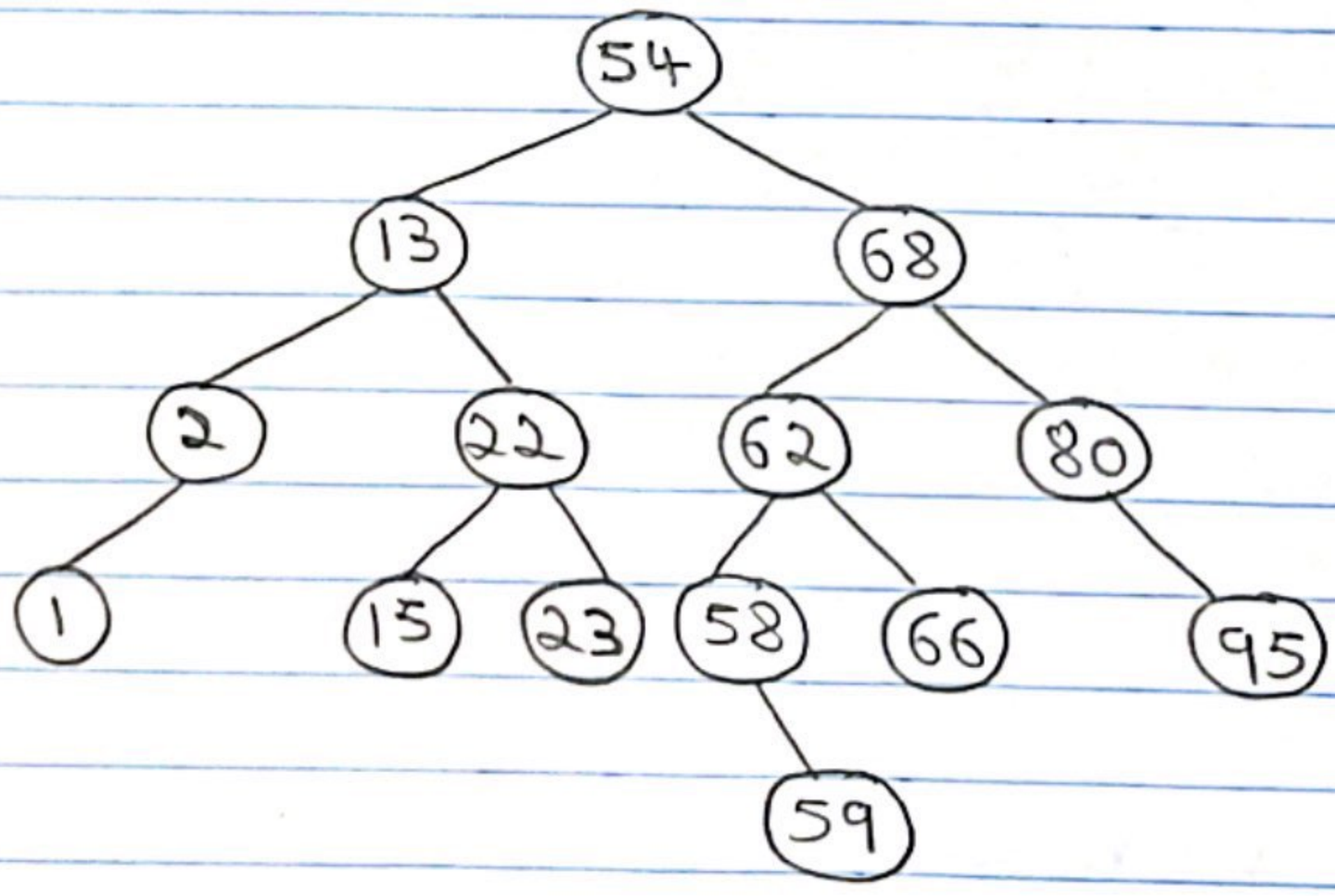
In each tree height is increased by 1 i.e.
Height of each tree is 5.

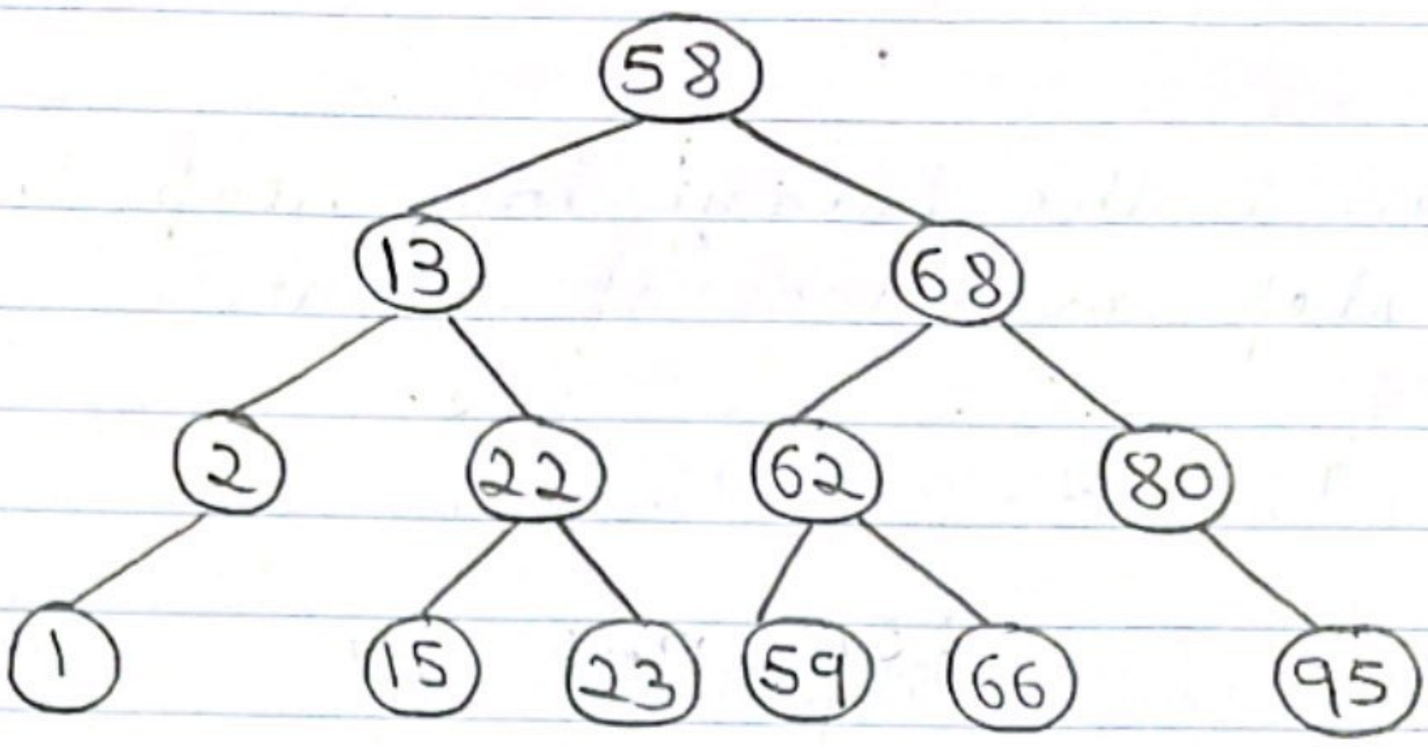
III) The four root numbers that will be removed are: 24, 54, 58, 59.

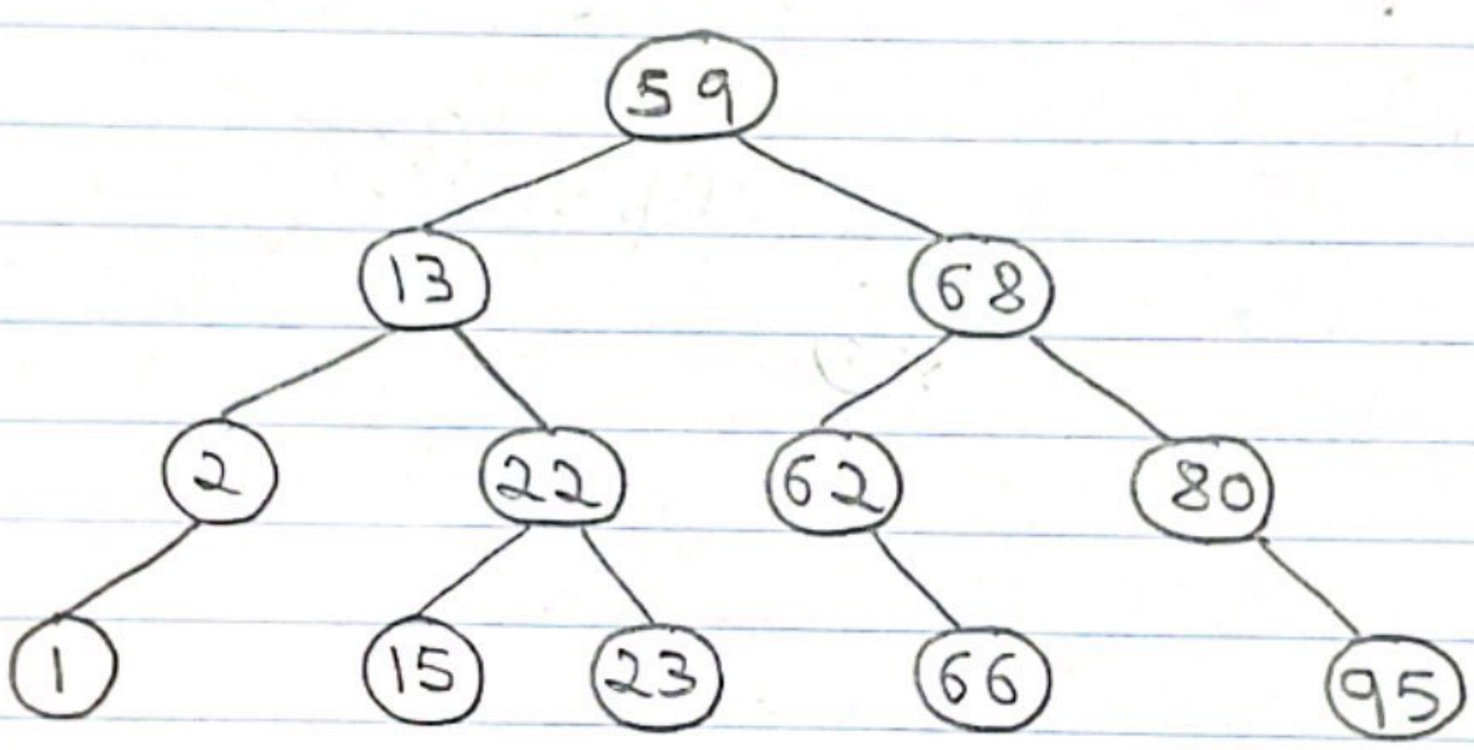
Below are the respective trees after removal.

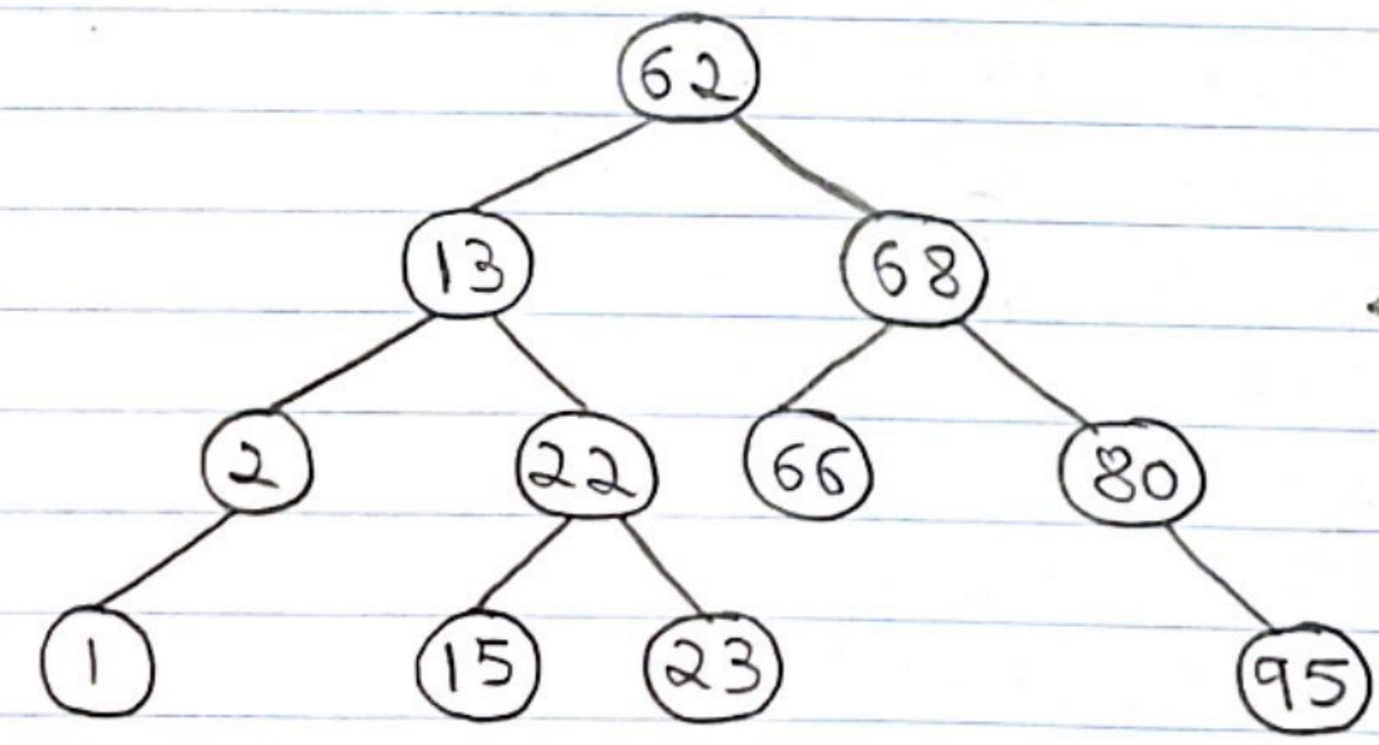
Tree from Part (I)











⇐ Final tree

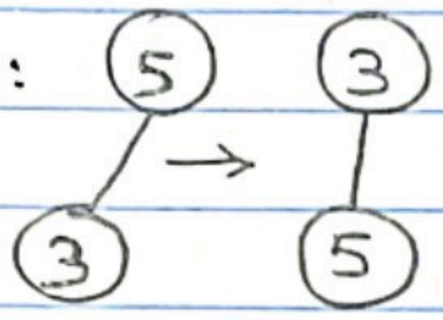
Q3)

Below is the binary min heap tree with step by step insertion of elements.

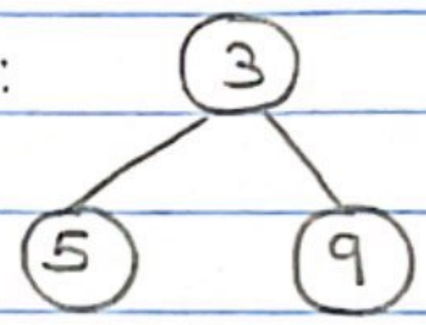
5, 3, 9, 7, 2, 4, 6, 1, 8

Step 1:

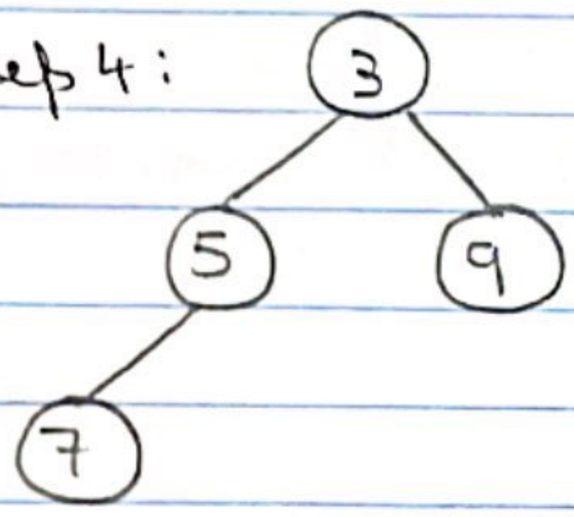
Step 2:



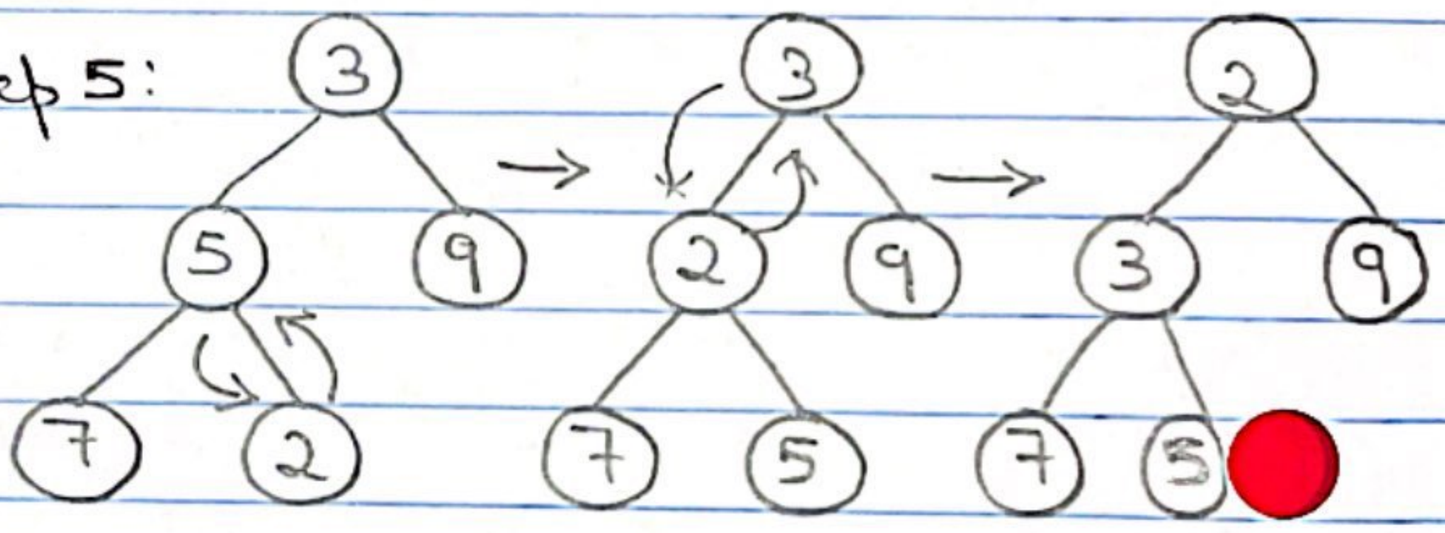
Step 3:



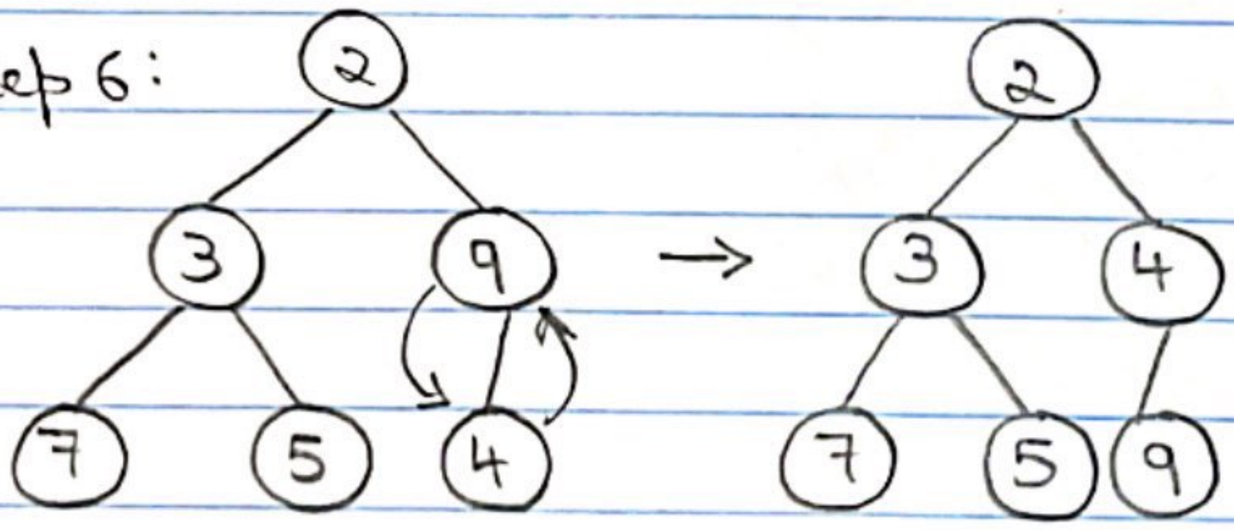
Step 4:



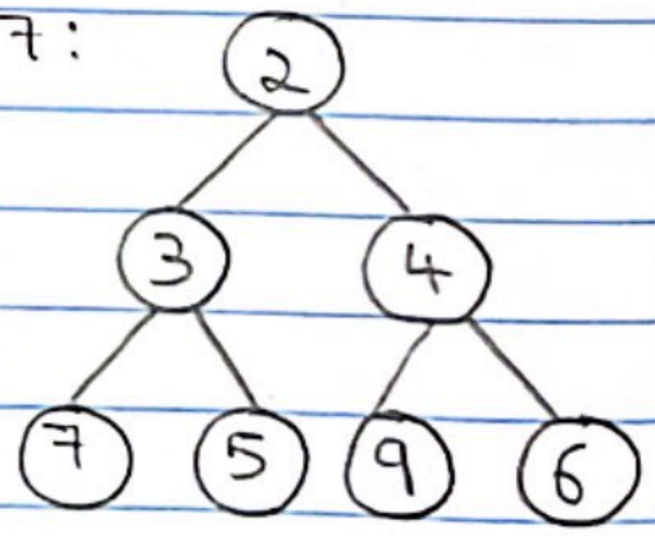
Step 5:



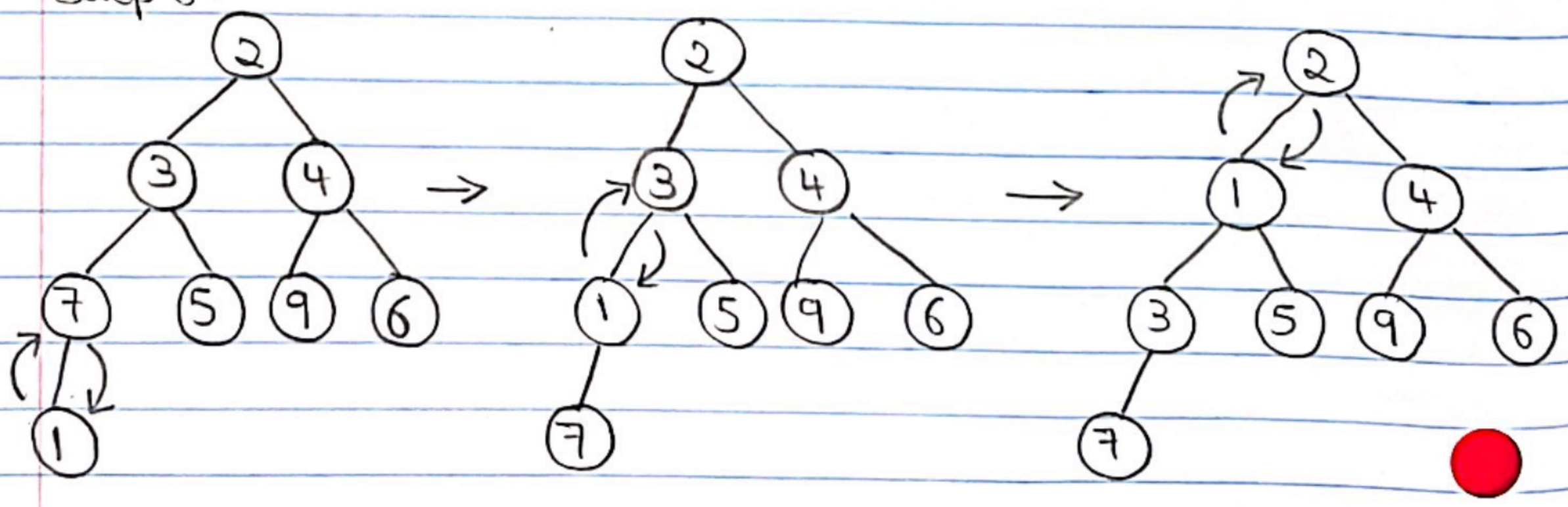
Step 6:



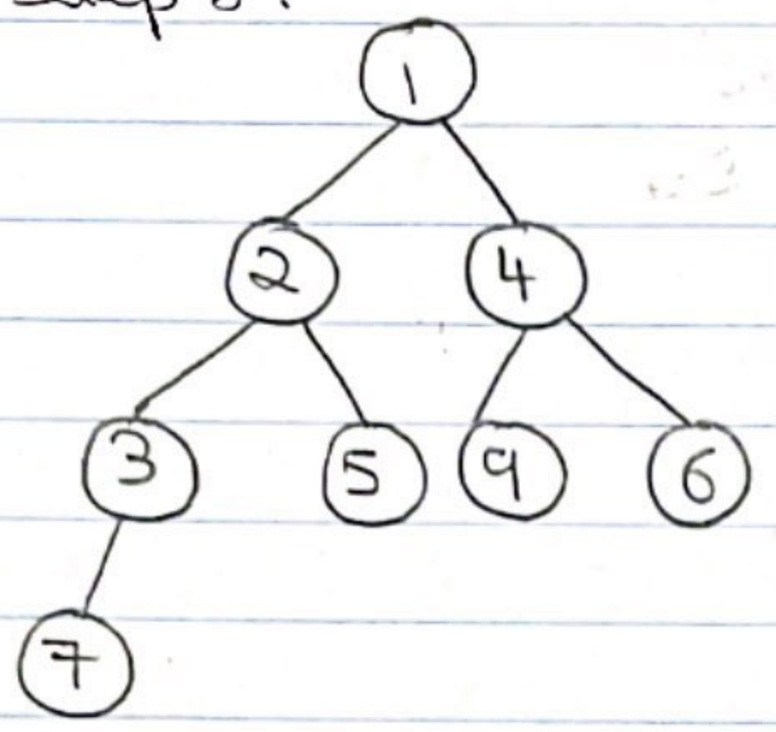
Step 7:



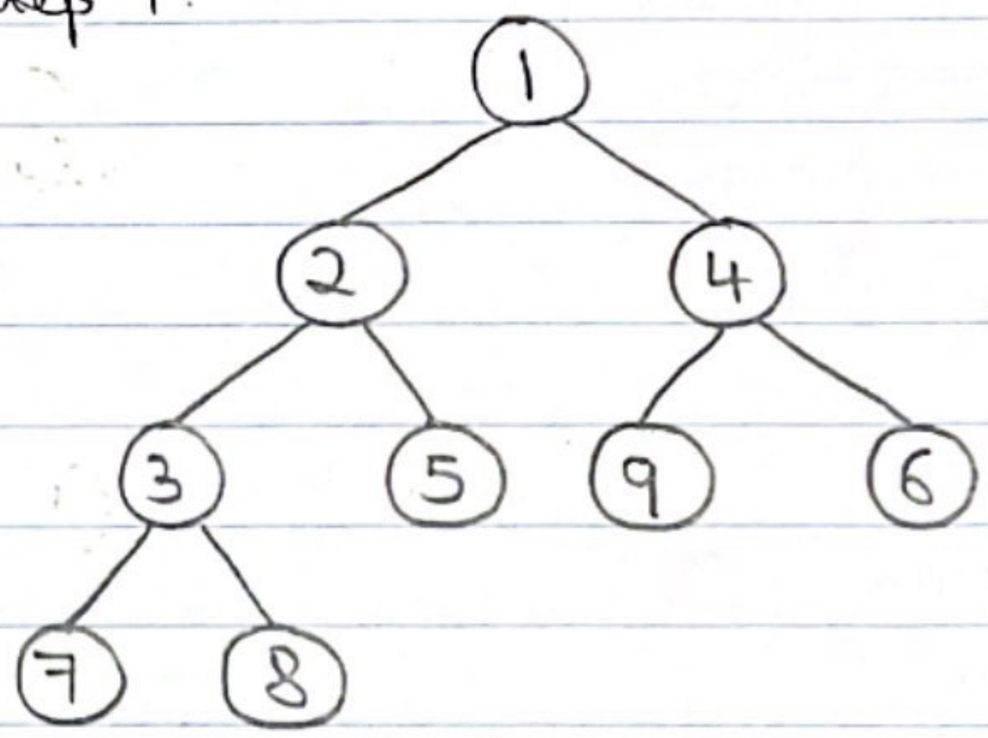
Step 8:



Step 8:



Step 9:



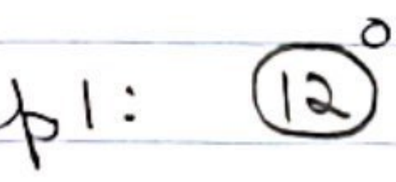
Level Order Traversal array is as below:

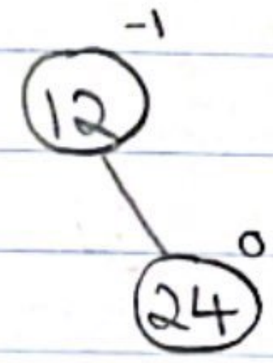
1, 2, 4, 3, 5, 9, 6, 7, 8

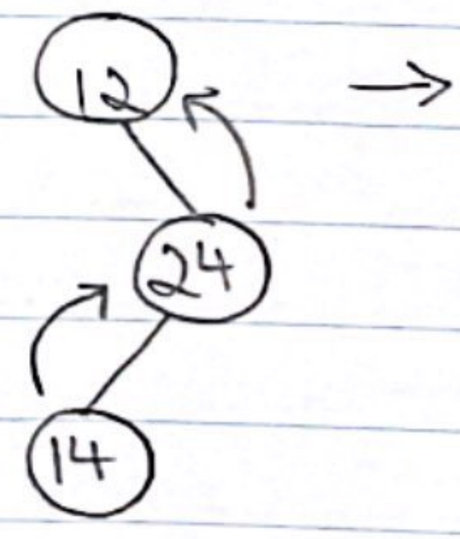
Q4>

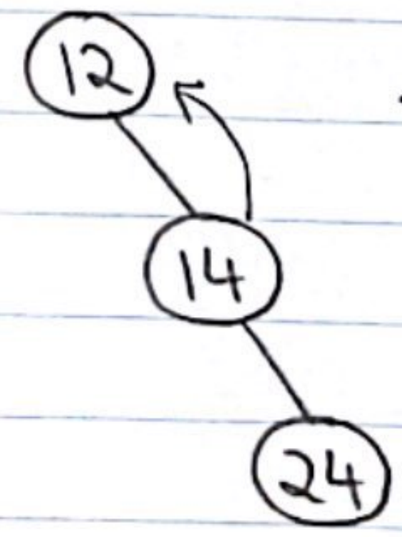
I> Below is the AVL tree created with given value.

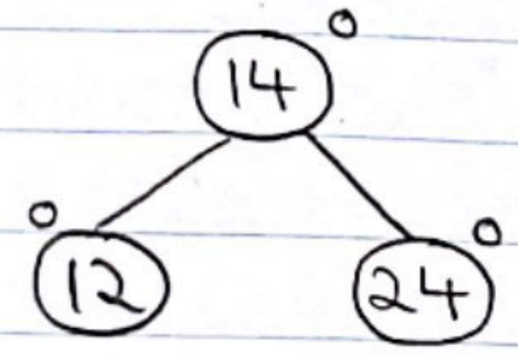
12, 24, 14, 27, 35, 17, 19, 22

Step 1: 

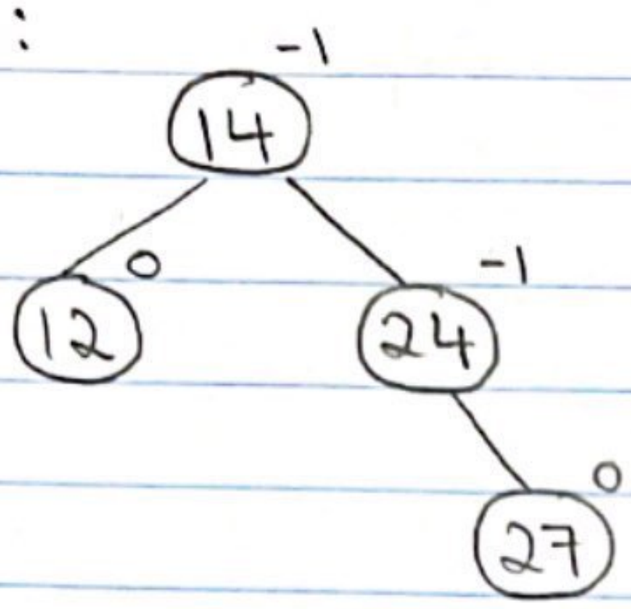
Step 2: 

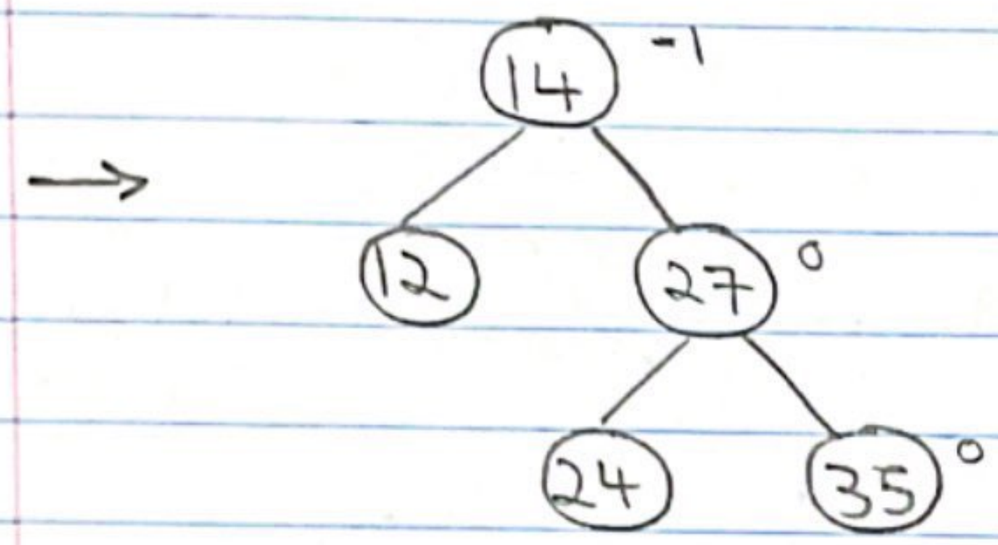
Step 3: 

Using RL rotation → 

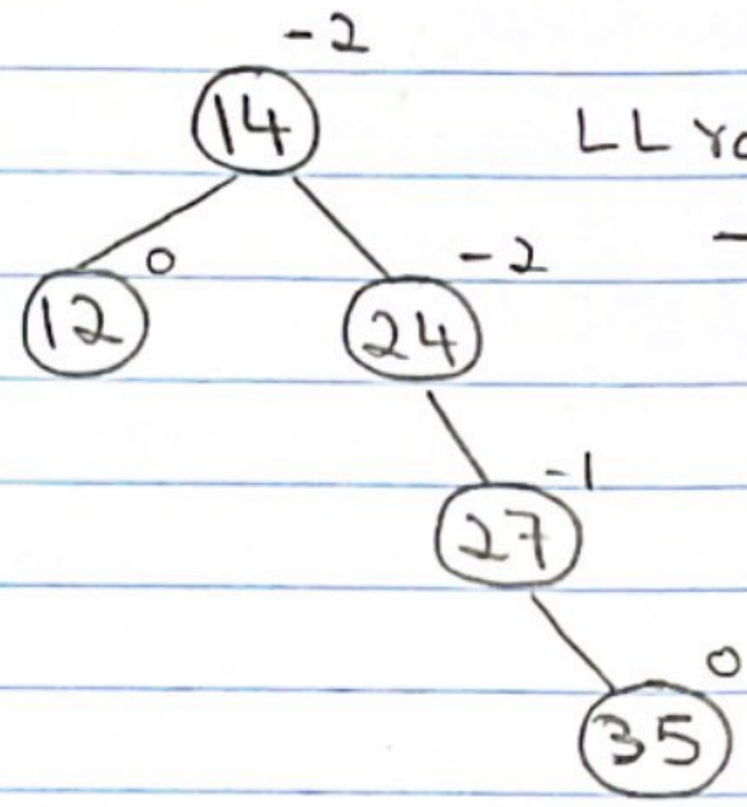
→ 

Step 4:



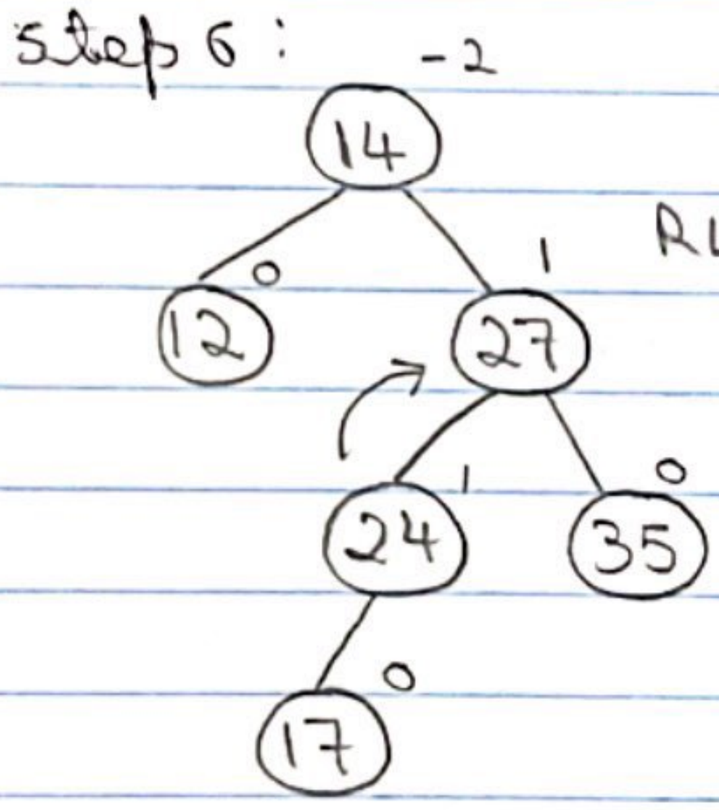


Step 5:

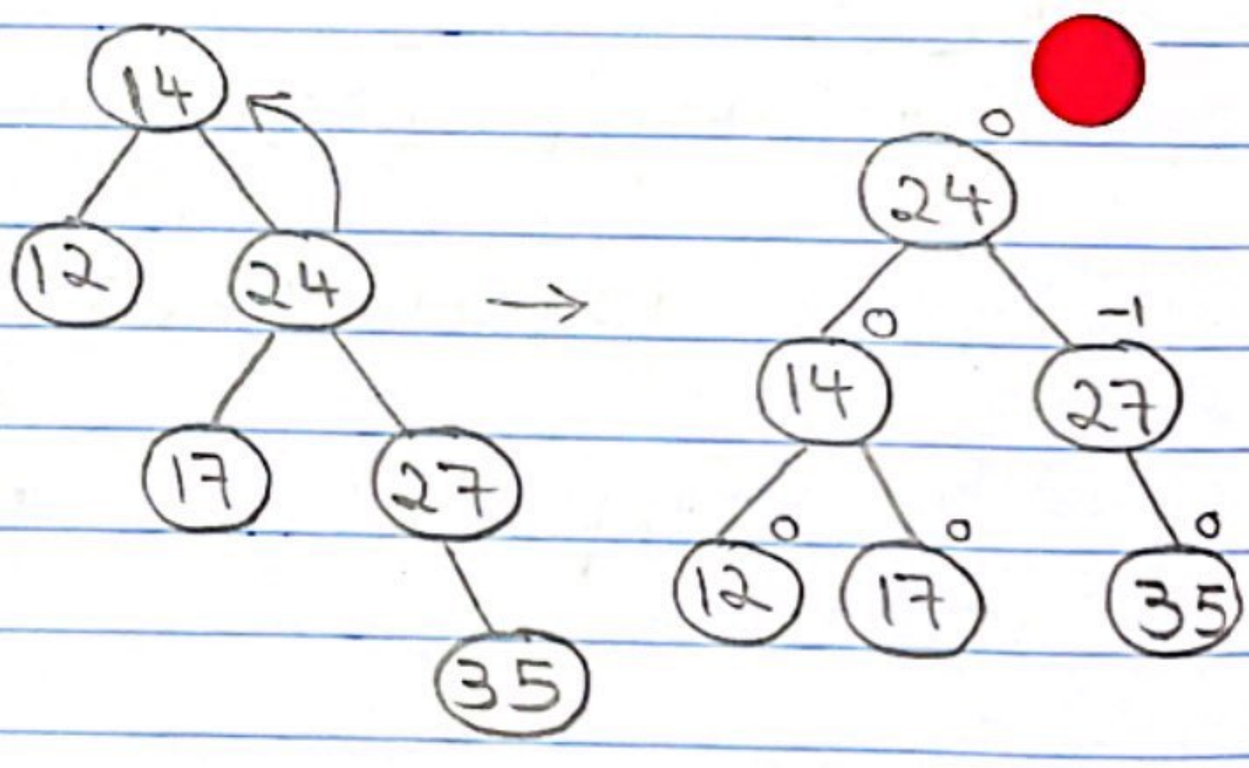


LL rotation.

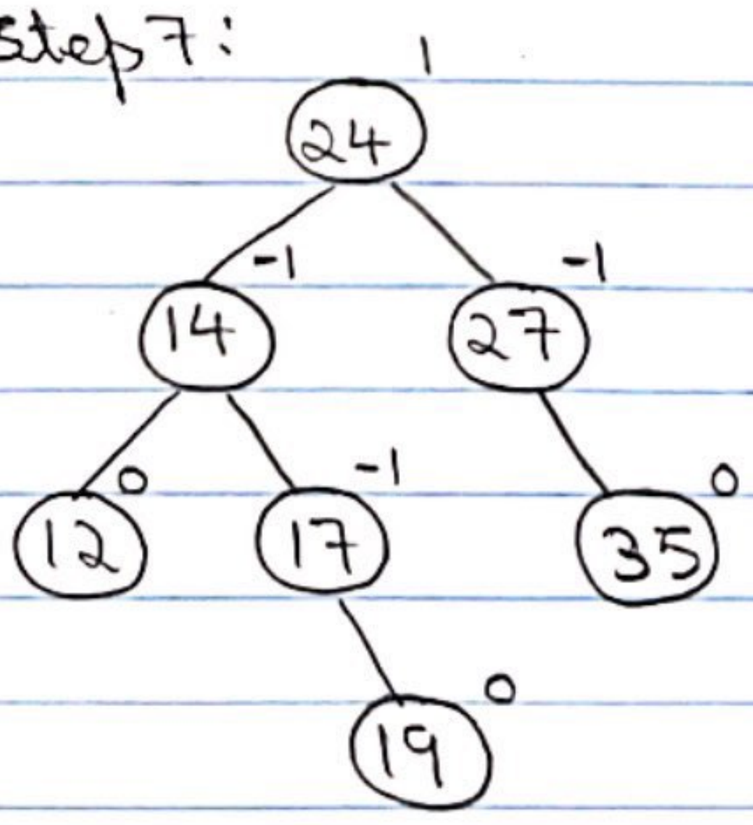
Step 6:



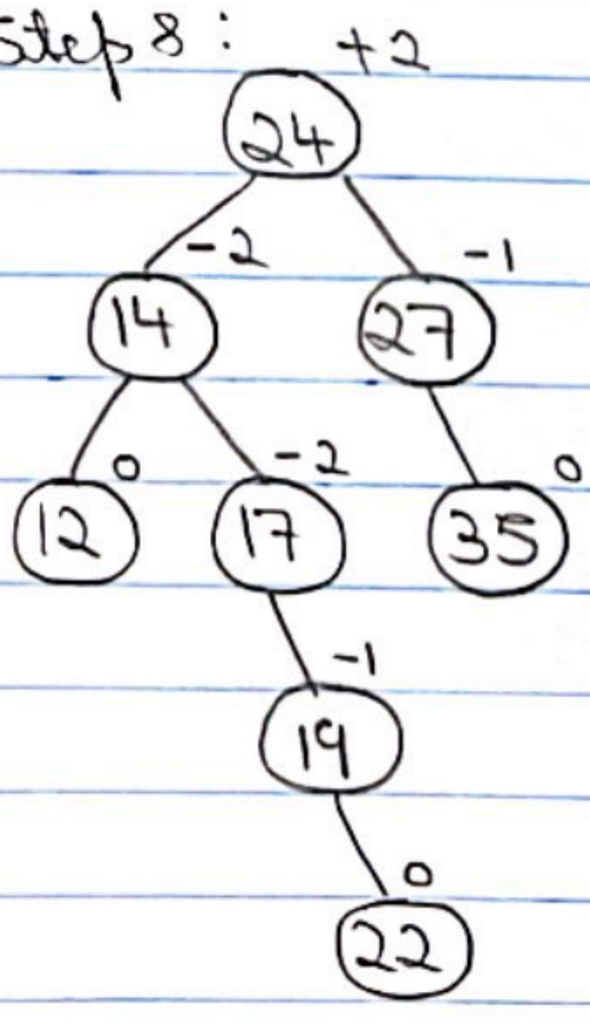
RL rotation



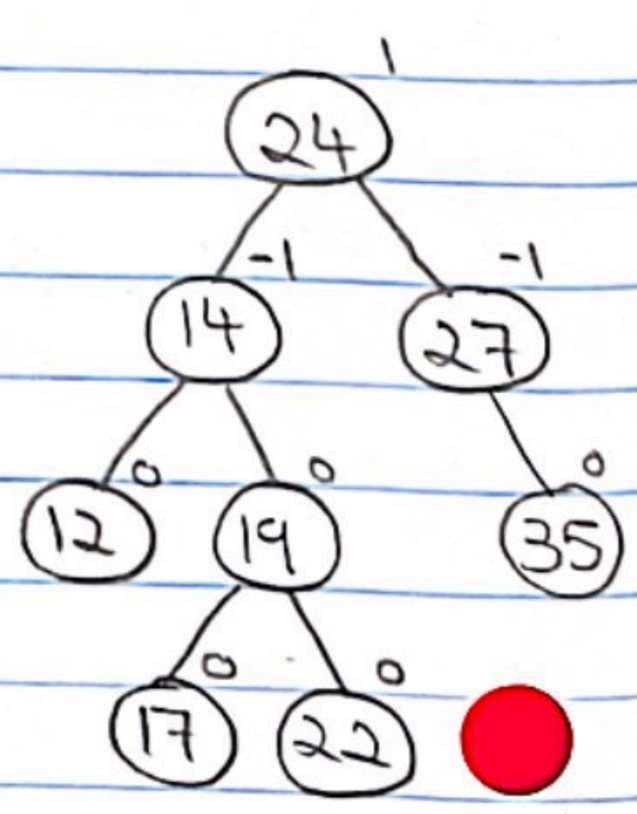
Step 7:



Step 8:

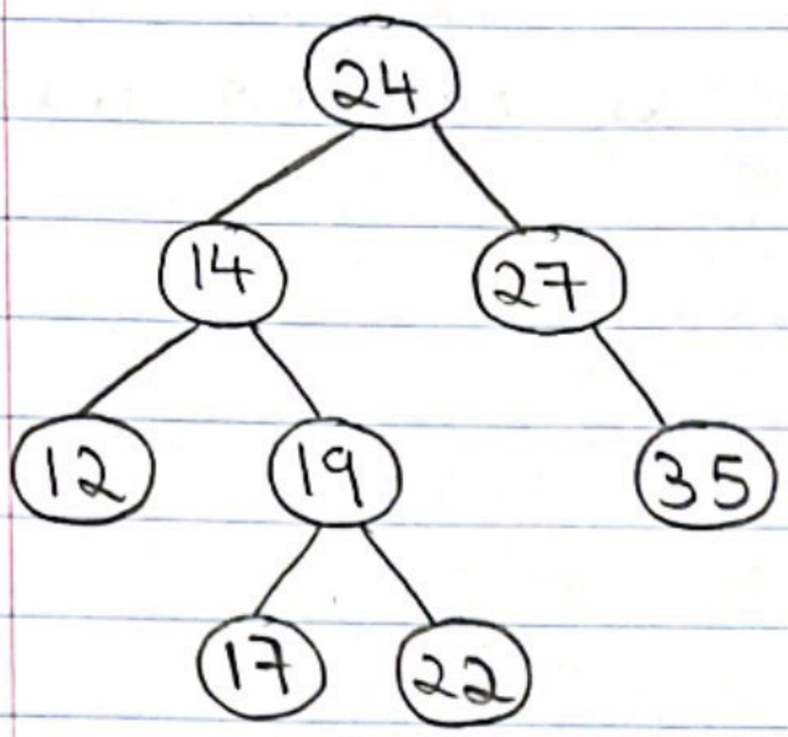


LL rotation

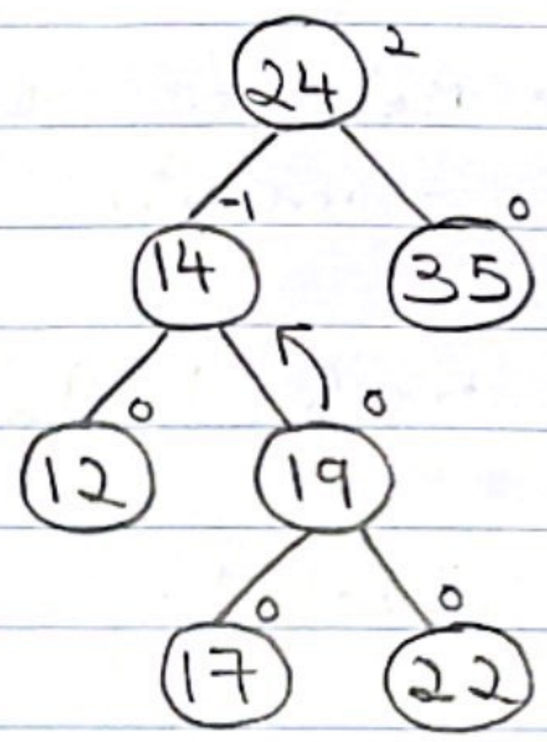


(Balaced AVL tree)

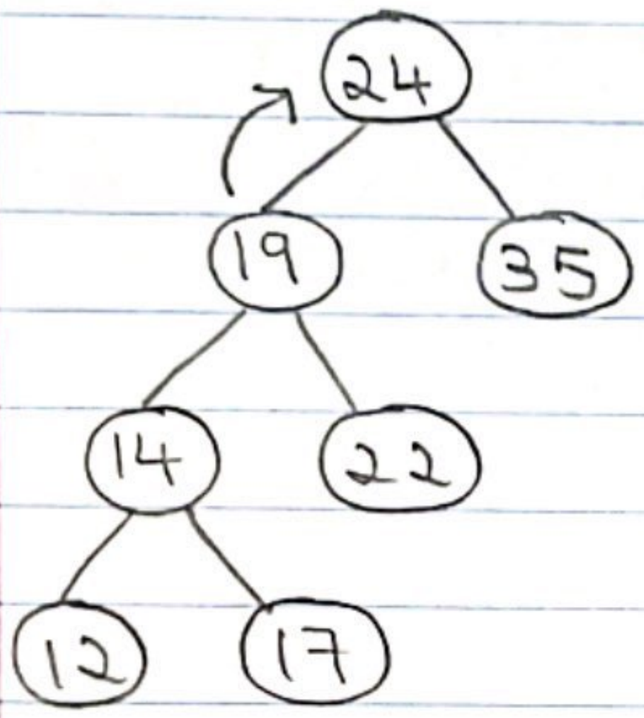
II:>

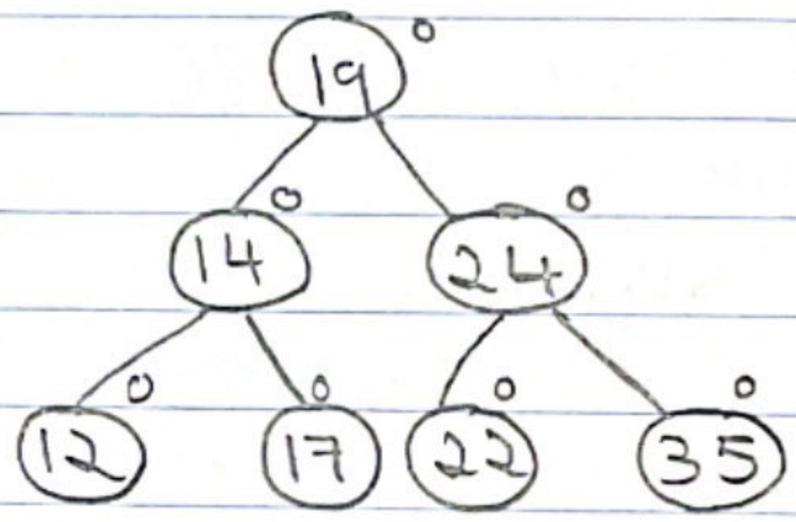


Delet 27 →



LR rotation →





(Final AVE tree after deleting element 27)

III:> The condition of AVL tree for each node is BST always have a balance factor of -1 or 0 or 1. Therefore maximum difference in heights between the leafs of a AVL tree is 1.

Q 5:>

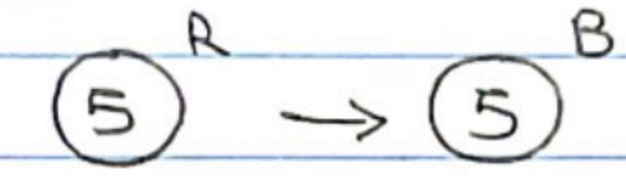
I:> The operations the could be performed in O(log n) time complexity by red-black tree.

a. Insertion
b. Deletion
c. Searching

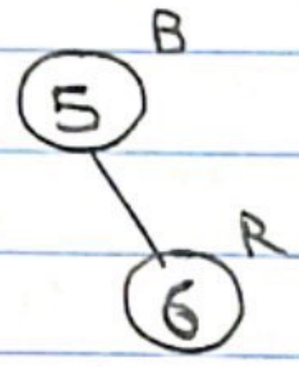
II) Below red black tree is created with given elements:

5, 6, 1, 9, 2, 4, 3, 8, 7

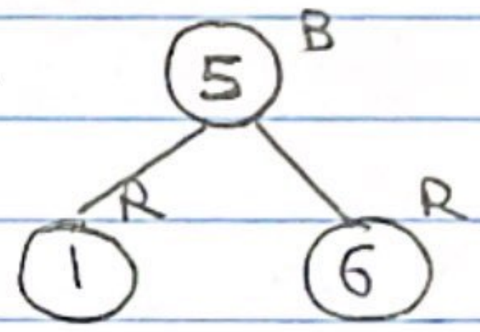
Step 1: Root node



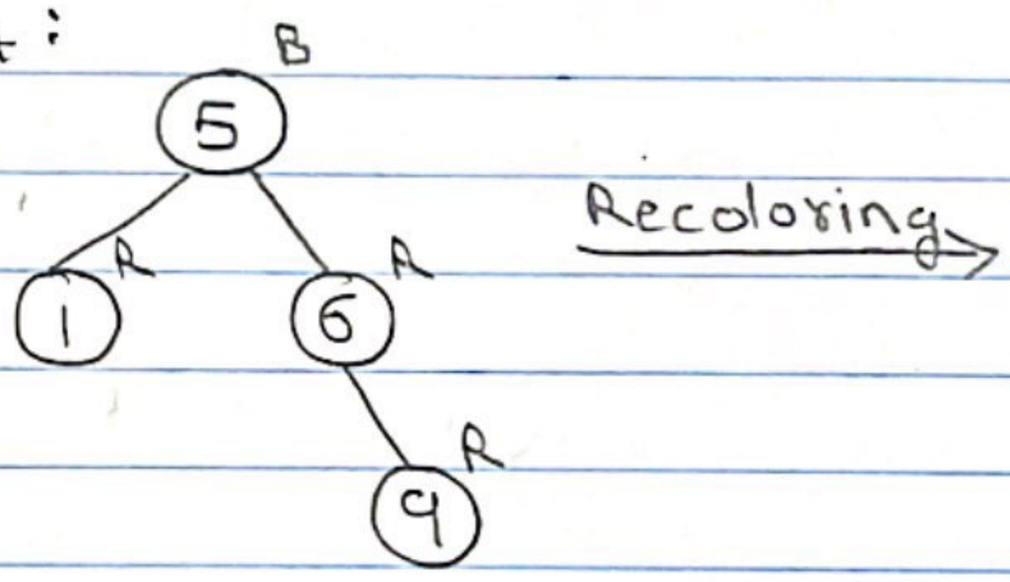
Step 2:

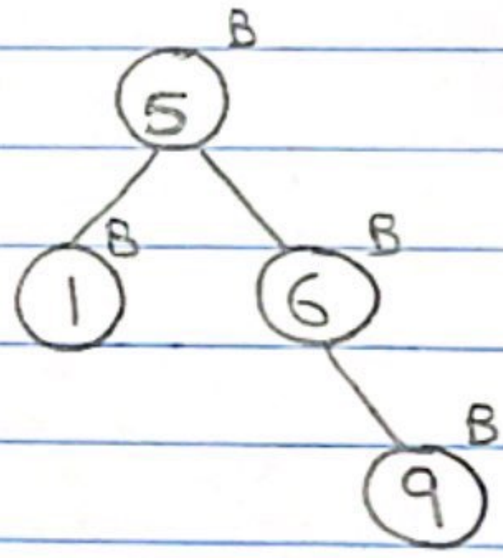


Step 3:

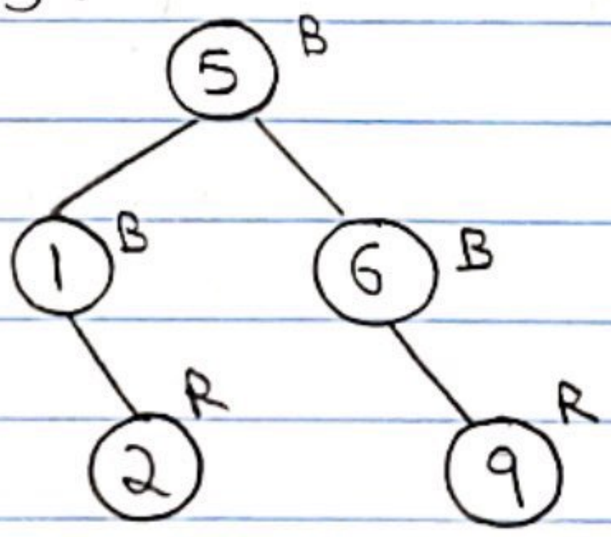


Step 4:

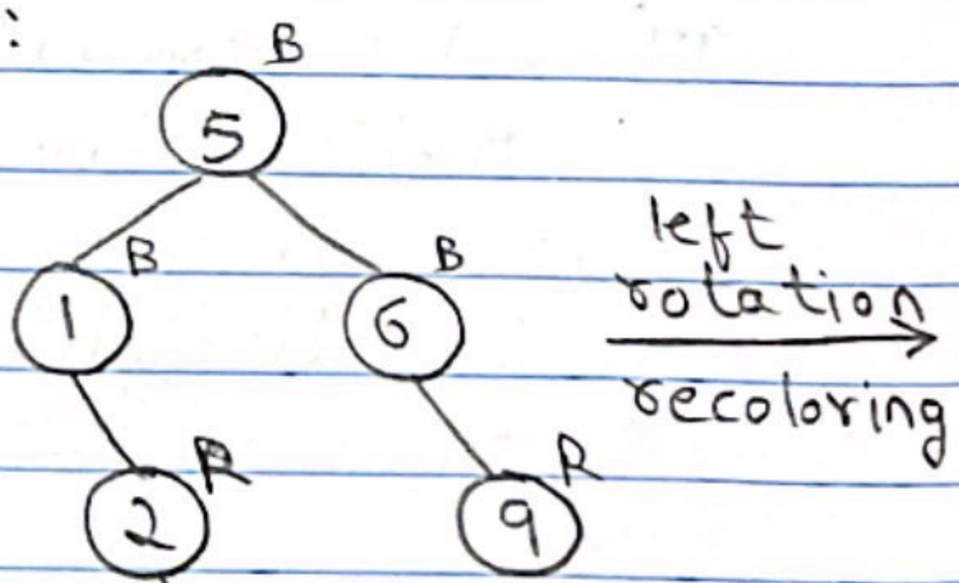


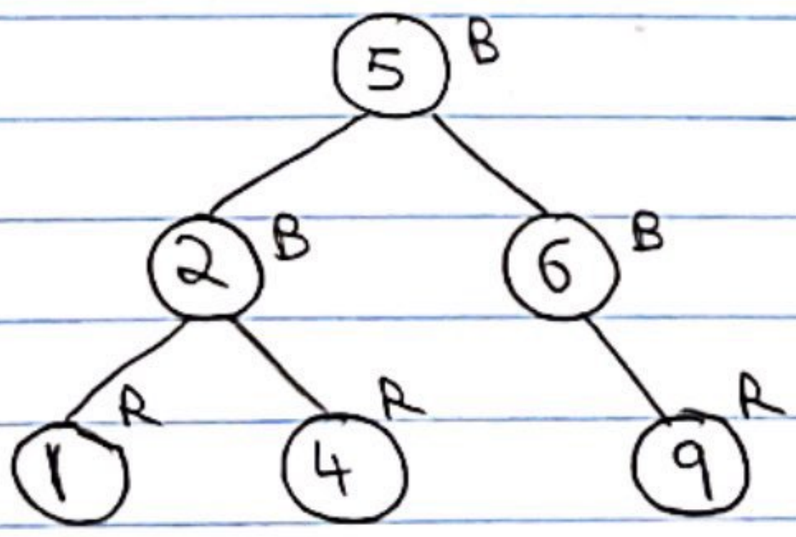


Step 5:

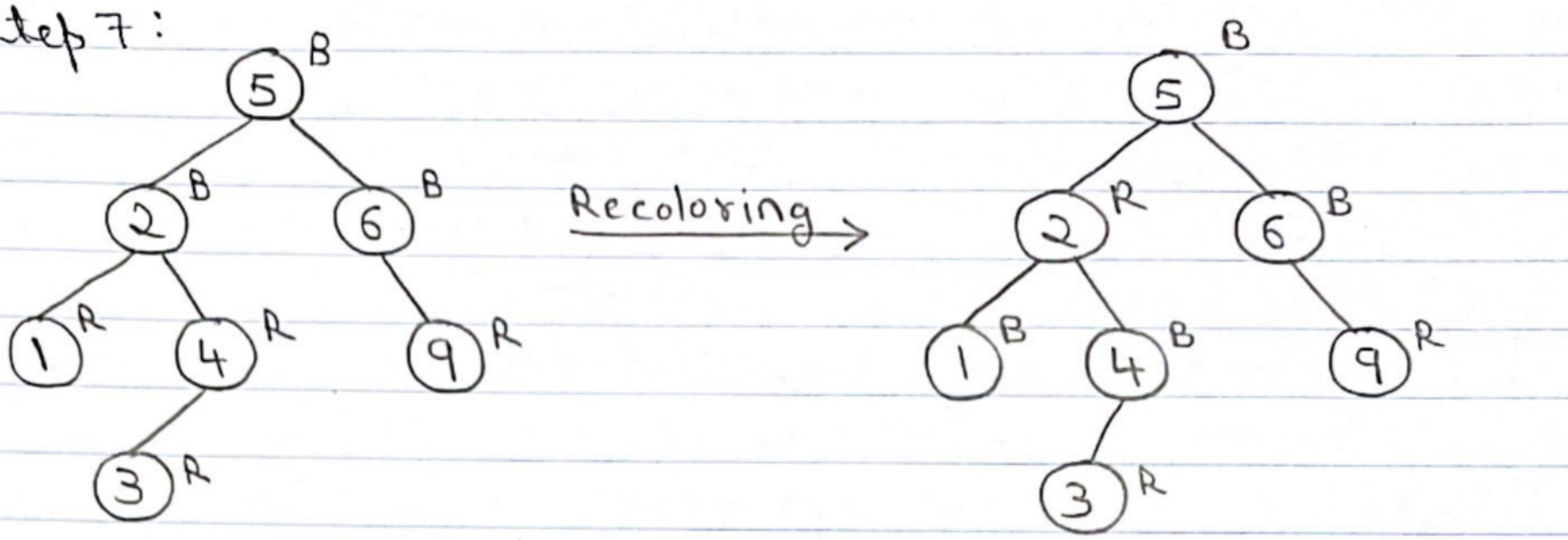


Step 6:

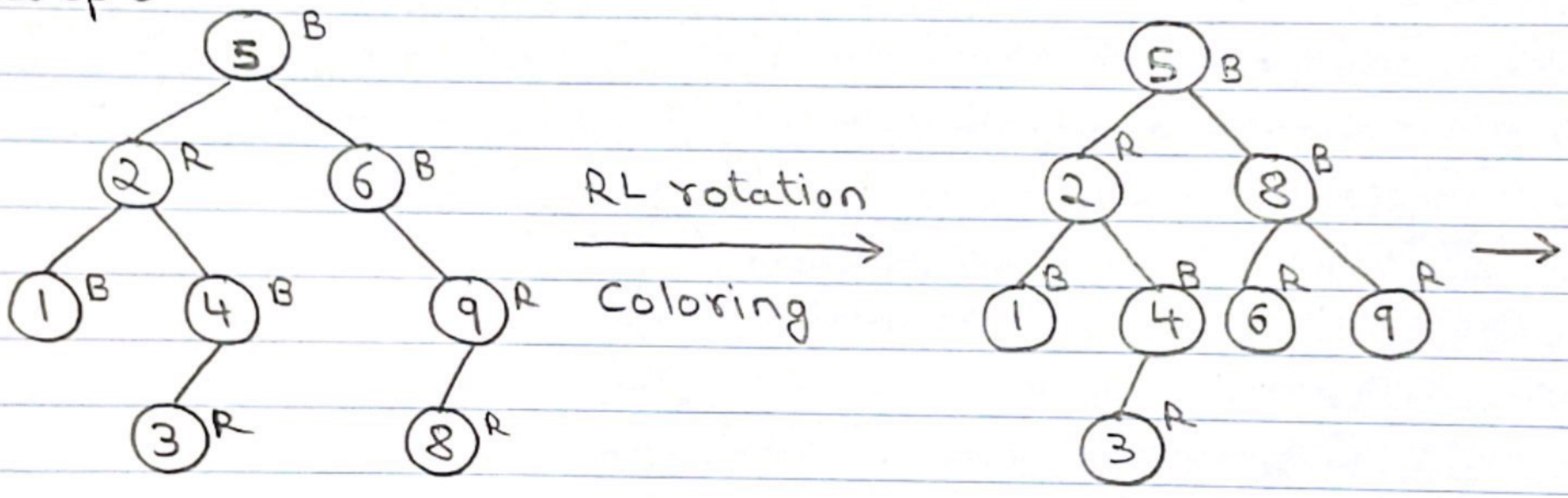


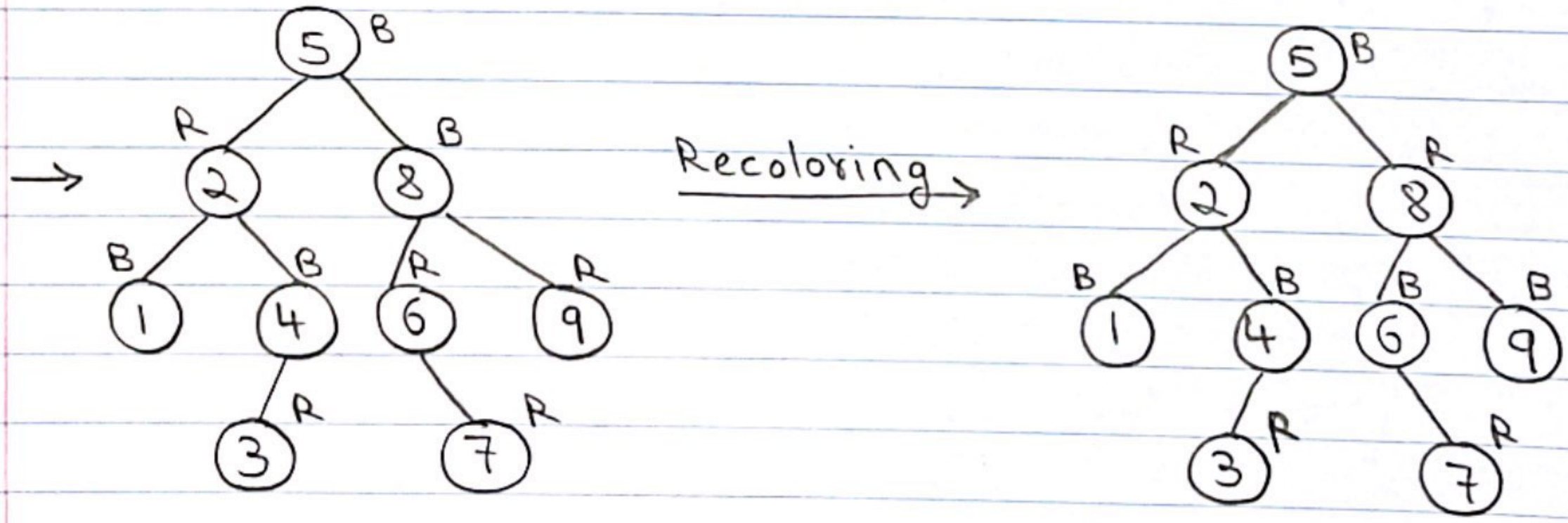


Step 7:



Step 8:





(Above tree is final Red-Black tree.)