PROBLEM

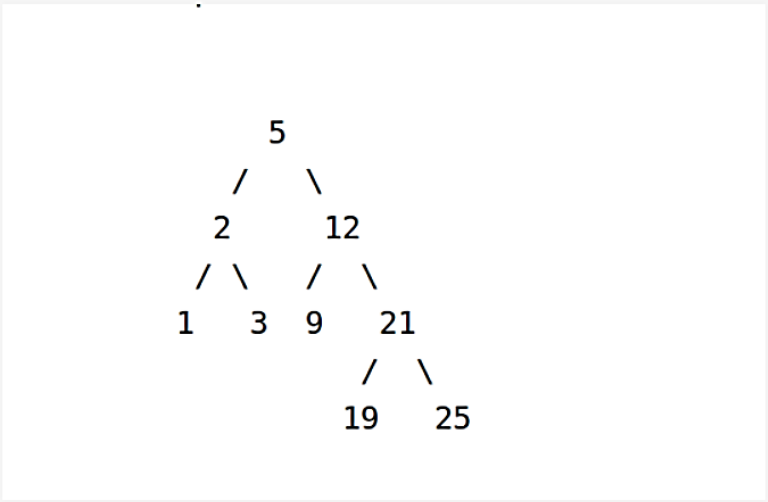
Closest Neighbour in BST

Easy Accuracy: 36.98% Submissions: 22K+ Points: 2

Given the **root of a binary search tree** and a number **n**, find the greatest number in the binary search tree that is less than or equal to **n**.

Example 1 :

Input :



n = 24

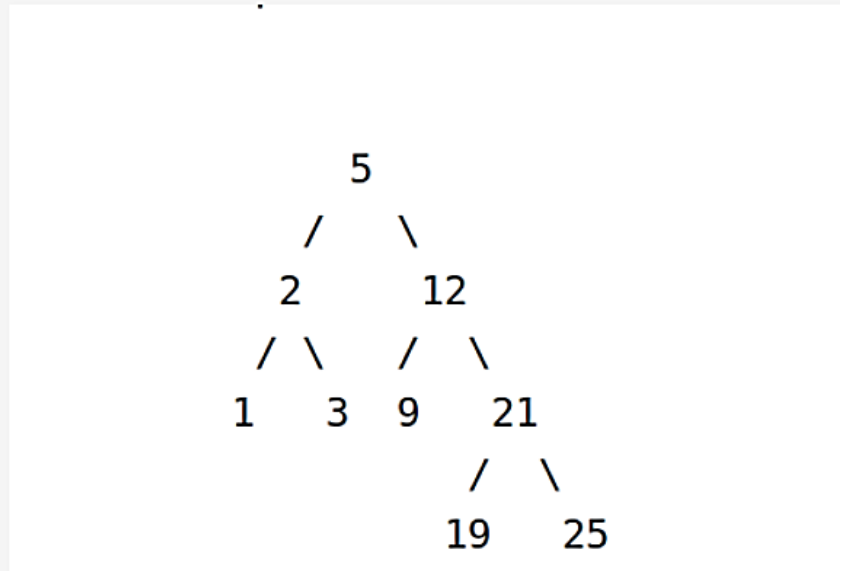
Output :

21

Explanation : The greatest element in the tree which is less than or equal to 24, is 21. (Searching will be like 5->12->21)

Example 2 :

Input :



n = 4
Output :
3
Explanation : The greatest element in the tree which is less than or equal to 4, is 3. (Searching will be like 5->2->3)

Your task :
You don't need to read input or print anything. Your task is to complete the function **findMaxForN()** which takes the **root** of the BST, and the integer **n** as input paramters and returns the **greatest element less than or equal** to **n** in the given BST.

Expected Time Complexity: O(Height of the BST)
Expected Auxiliary Space: O(Height of the BST).

Constraints:
$1 <= n <= 10^3$

CODE

```
#User function Template for python3


'''
class Node:
    """ Class Node """
    def _init_(self, value):
        self.left = None
        self.key = value
        self.right = None
'''

class Solution:
    def findMaxForN(self, root, n):
        #Print the value of the element if it exists otherwise print -1.
        temp = root
        ans = -1
        while temp:
            if temp.key<=n:
                ans = max(ans, temp.key)

            if temp.key>n:
                temp = temp.left
            else:
                temp = temp.right
        #code here

        return ans
```

EXPLANATION

First, we have a class Node representing a node in a binary tree. It has three attributes: left, key, and right.

```
class Node:
    """ Class Node """
    def _init_(self, value):
        self.left = None
        self.key = value
        self.right = None
```

In this class:

left points to the left child node.
key stores the value of the node.
right points to the right child node.

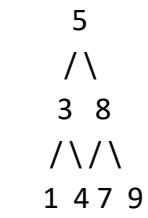
Next, we have the Solution class, which contains a method findMaxForN:

```
class Solution:
    def findMaxForN(self, root, n):
        temp = root
        ans = -1
        while temp:
            if temp.key <= n:
                ans = max(ans, temp.key)

            if temp.key > n:
                temp = temp.left
            else:
                temp = temp.right

        return ans
```

Explanation of findMaxForN method:

It takes two arguments: root, which is the root node of the binary tree, and n, which is the target value.
It initializes temp with the root node and ans with -1. temp is used to traverse the tree, and ans is used to store the maximum value found so far.
It enters a while loop that continues until temp becomes None, i.e., until we reach a leaf node.
Inside the loop, it checks if the value of the current node (temp.key) is less than or equal to the target value n. If it is, it updates ans to the maximum of ans and the current node's value.
If the current node's value is greater than n, it moves to the left child of the current node (temp = temp.left). Otherwise, it moves to the right child (temp = temp.right).

Finally, it returns the maximum value found (ans) after traversing the tree.
Now, let's visualize a simple binary tree and how the algorithm works:



Suppose we want to find the maximum value less than or equal to 6. We start from the root:

temp is initially at node 5. Since 5 <= 6, ans is updated to 5.
Move to the right child (temp = 8). Since 8 > 6, move to the left child (temp = 7). Now, 7 <= 6, update ans to 7.
Move to the left child of 7 (which is None). The loop ends.
Return ans which is 7.