Title: Trees

Author: Pınar Yücel

ID: 21802188

Assignment: 2

Question 1

Prefix (Preorder): - x x A B - + C D E / F + G H

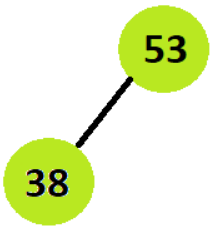
Infix (Inorder): A x B x C + D - E - F / G + H

Postfix (Postorder): A B x C D + E - x F G H + / -

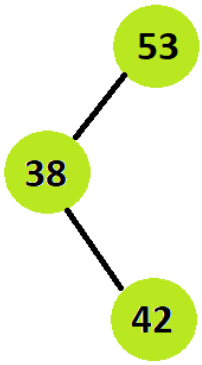
Question 2

After insertion (53)

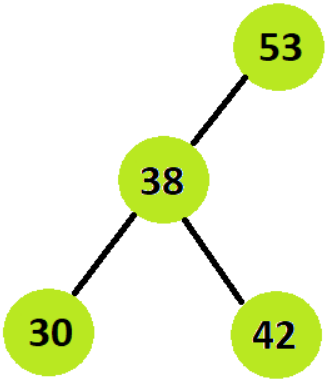
After insertion (38)



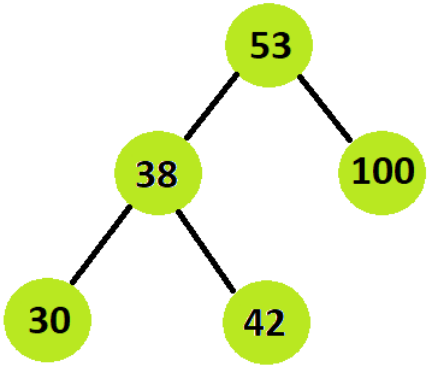
After insertion (42)



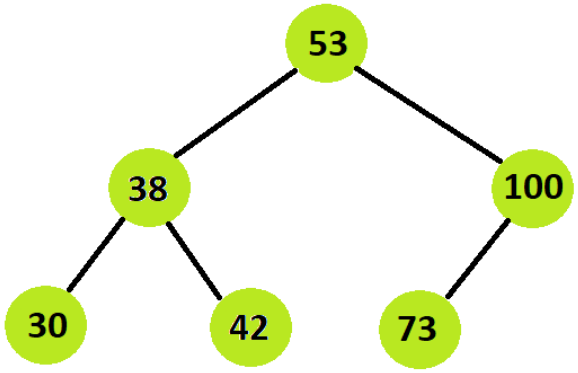
After insertion (30)



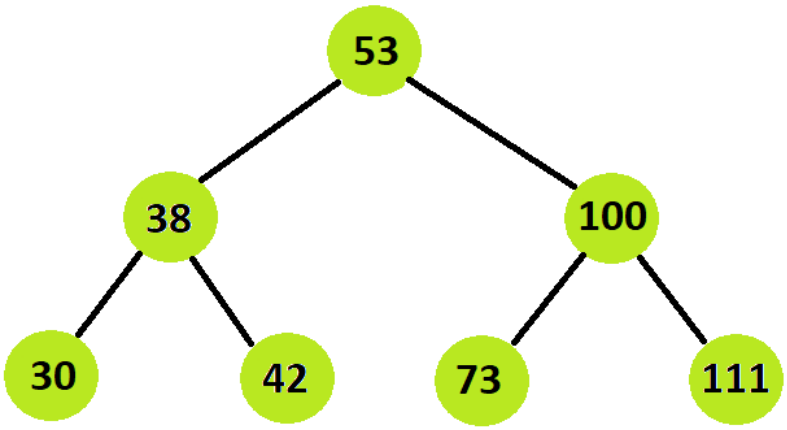
After insertion (100)



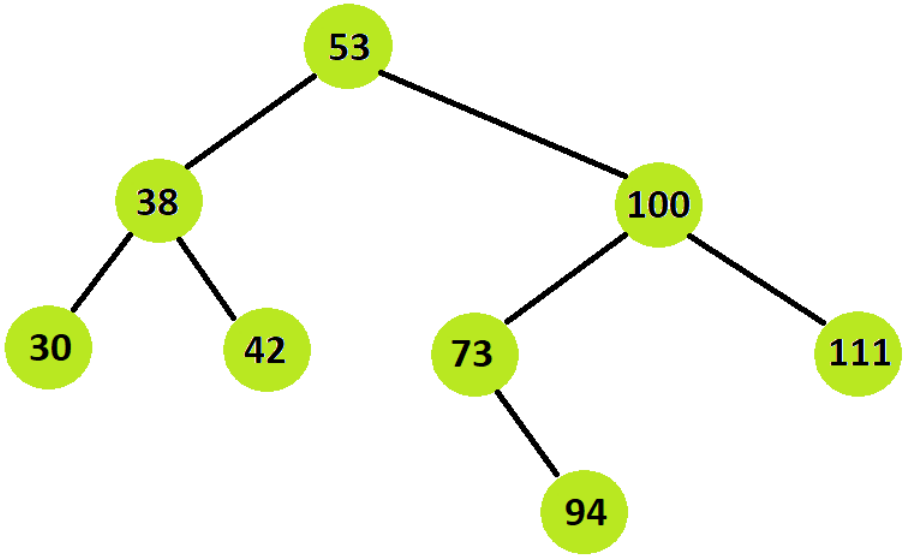
After insertion (73)



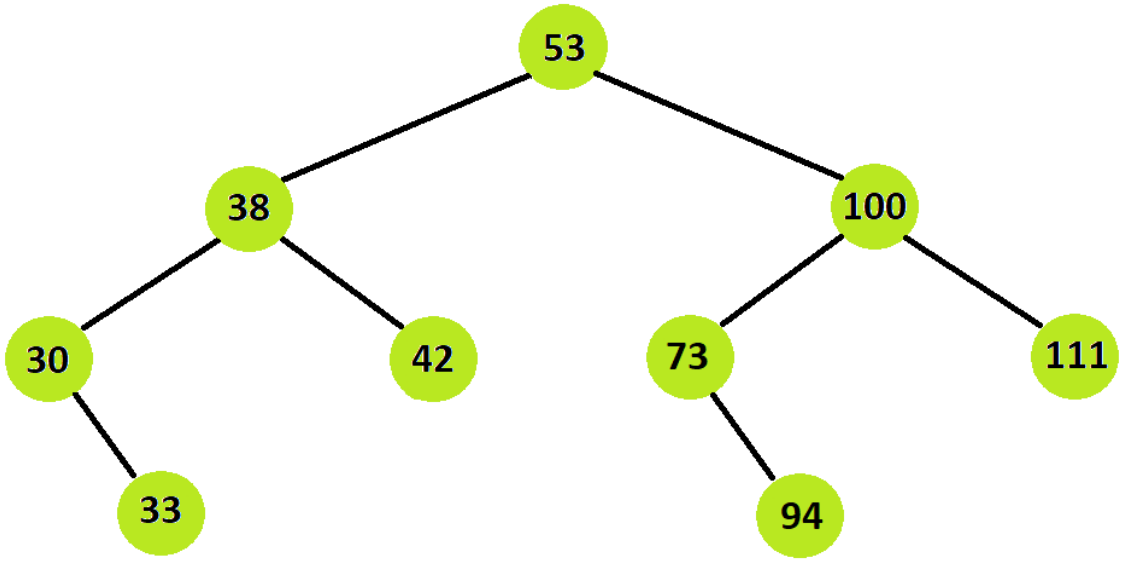
After insertion (111)



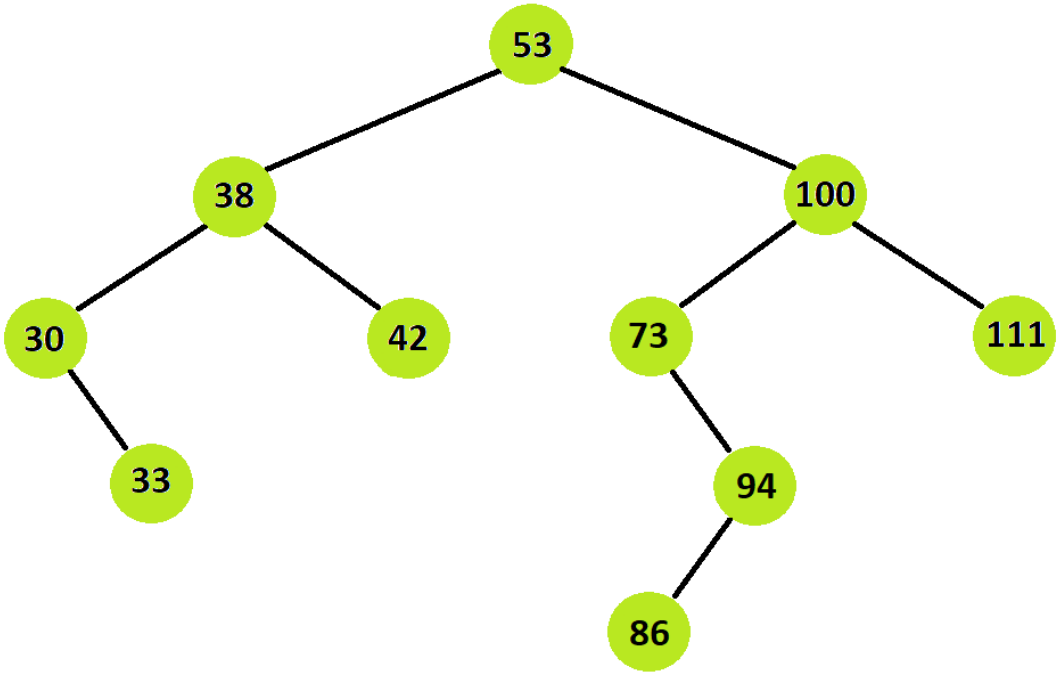
After insertion (94)



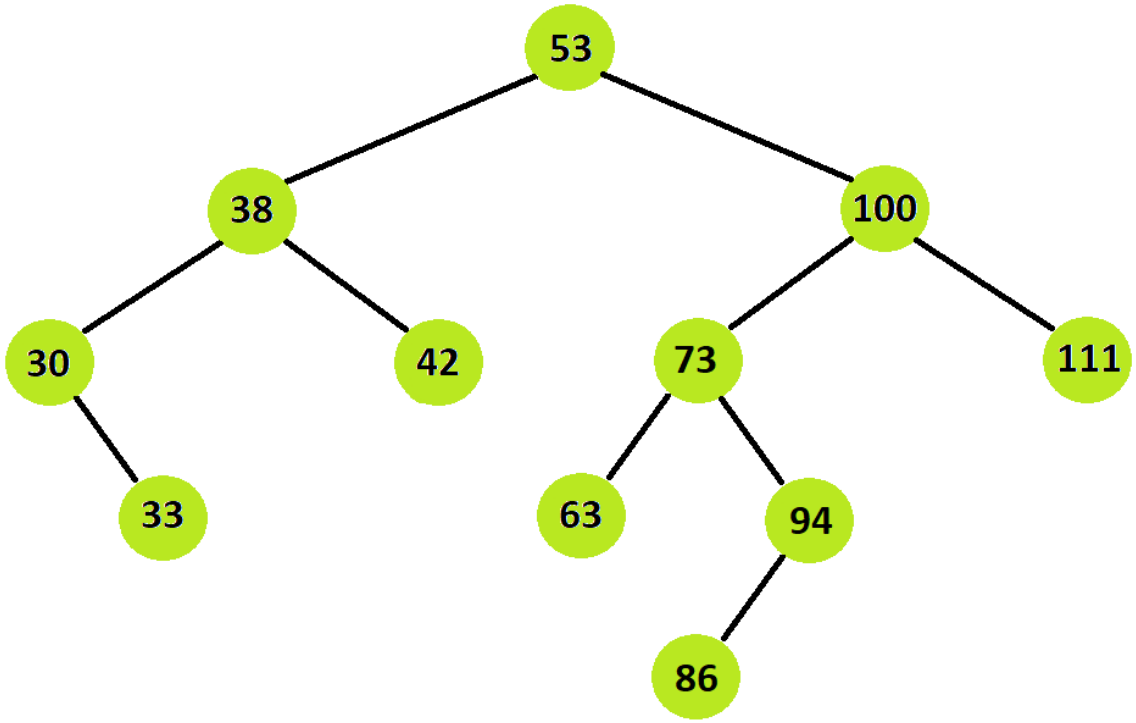
After insertion (33)



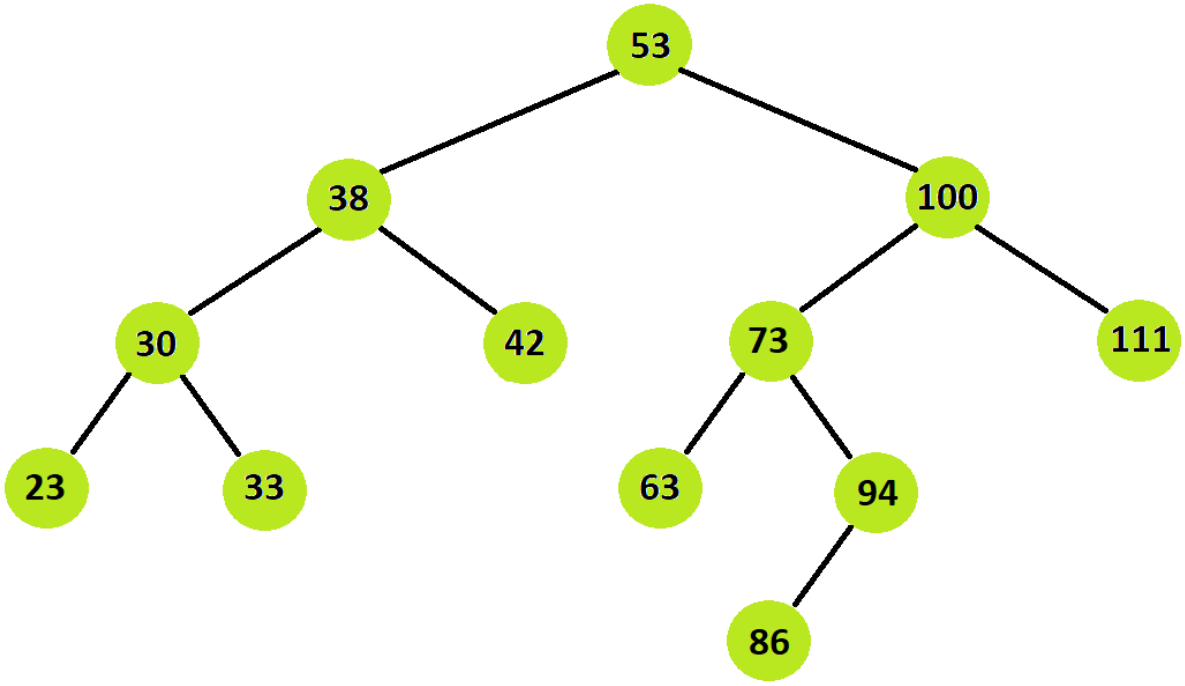
After insertion (86)



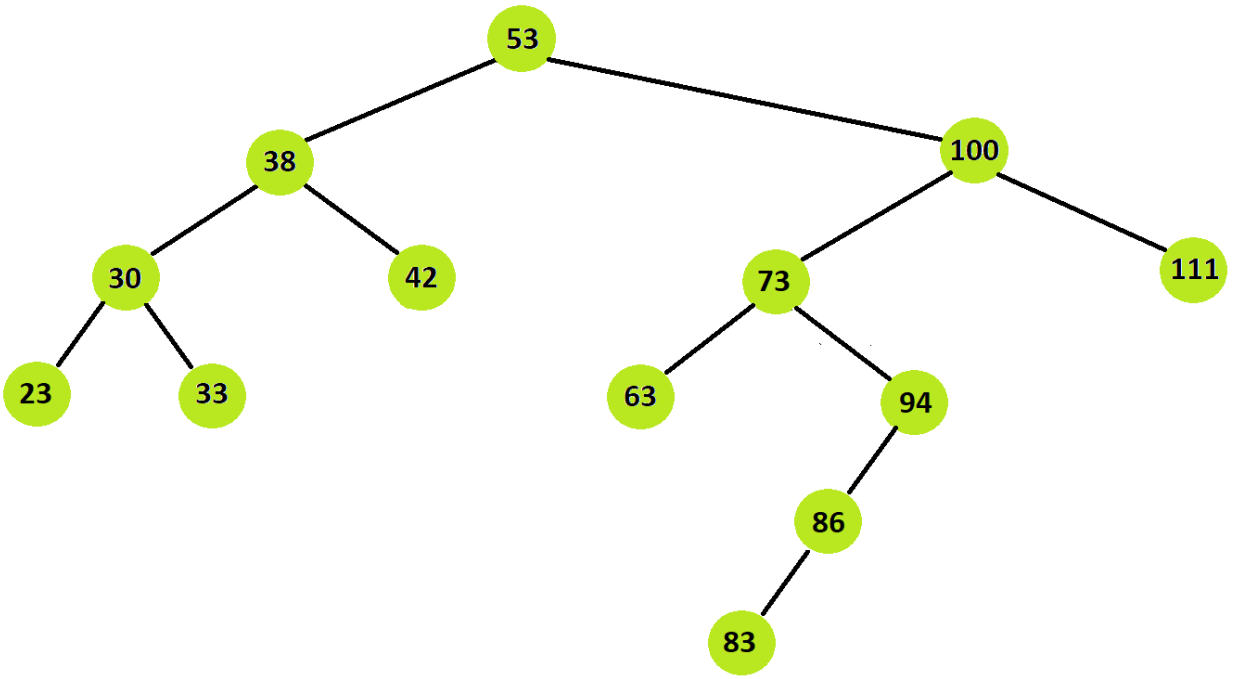
After insertion (63)



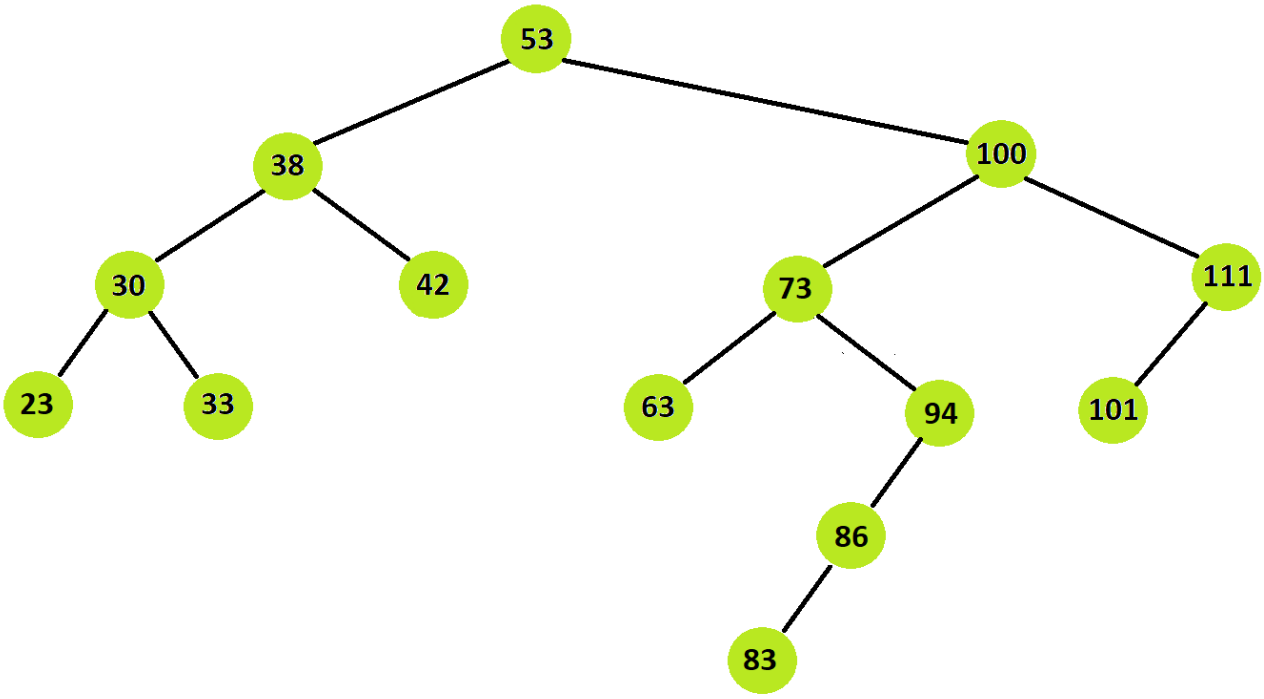
After insertion (23)



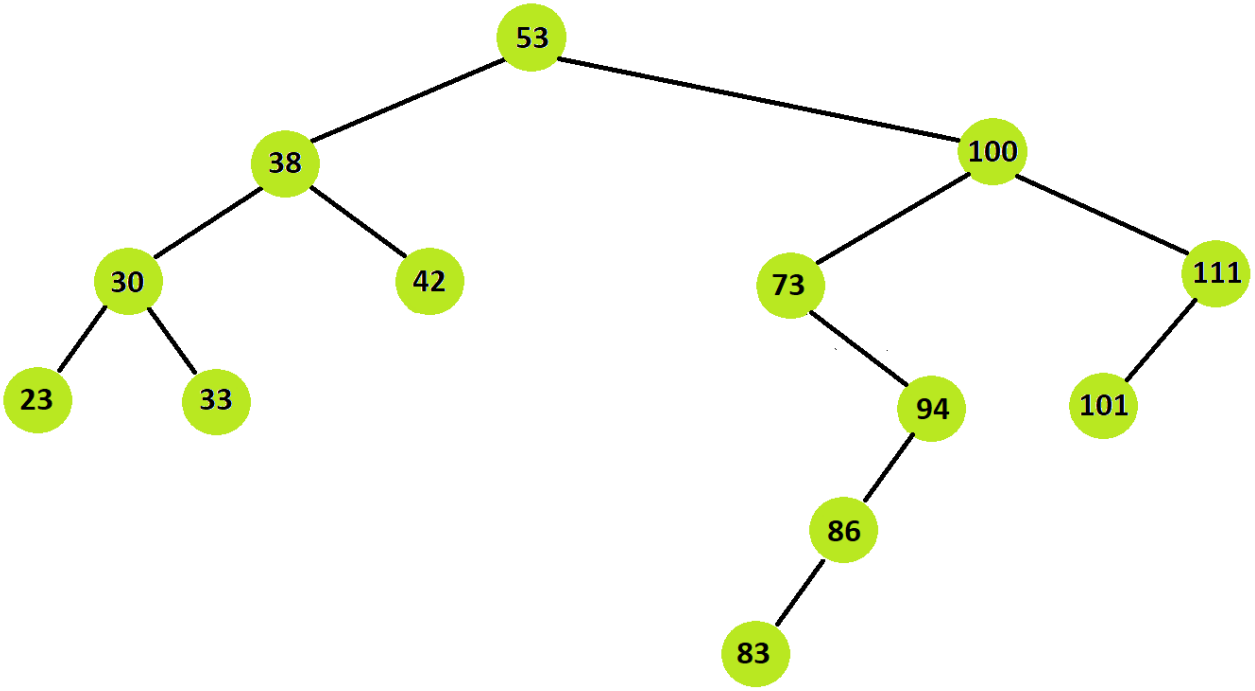
After insertion (83)



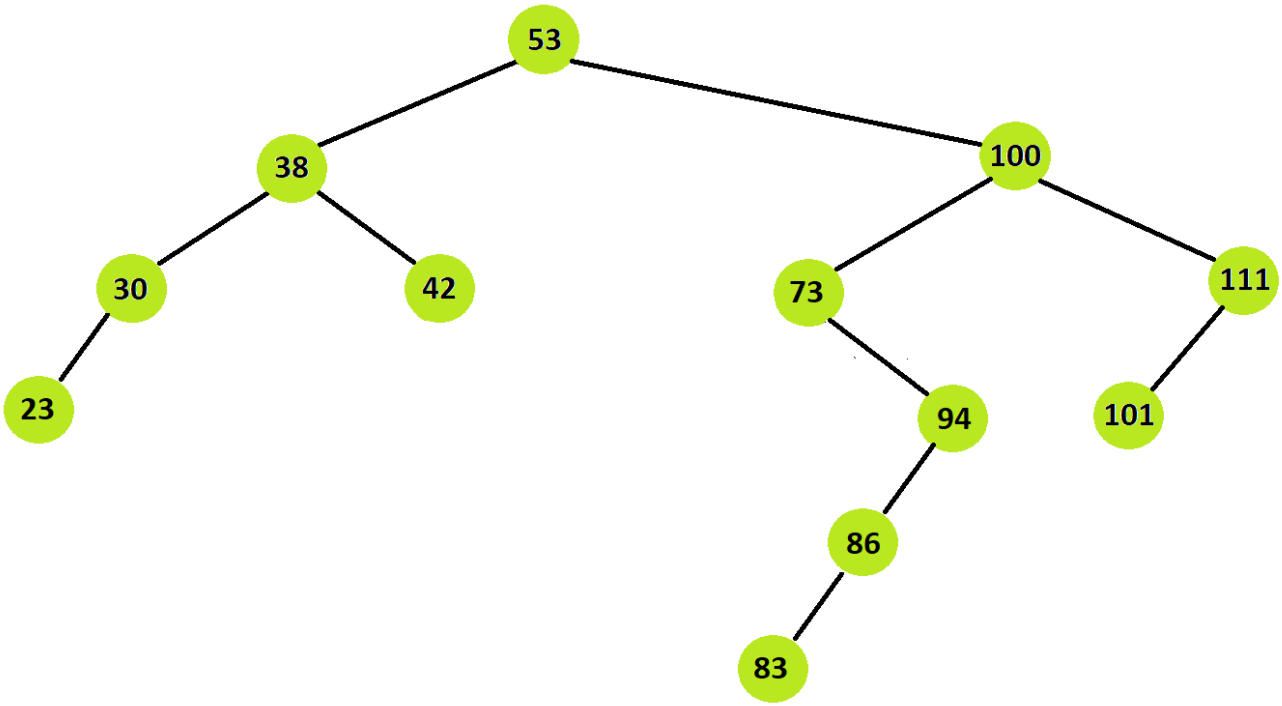
After insertion (101)



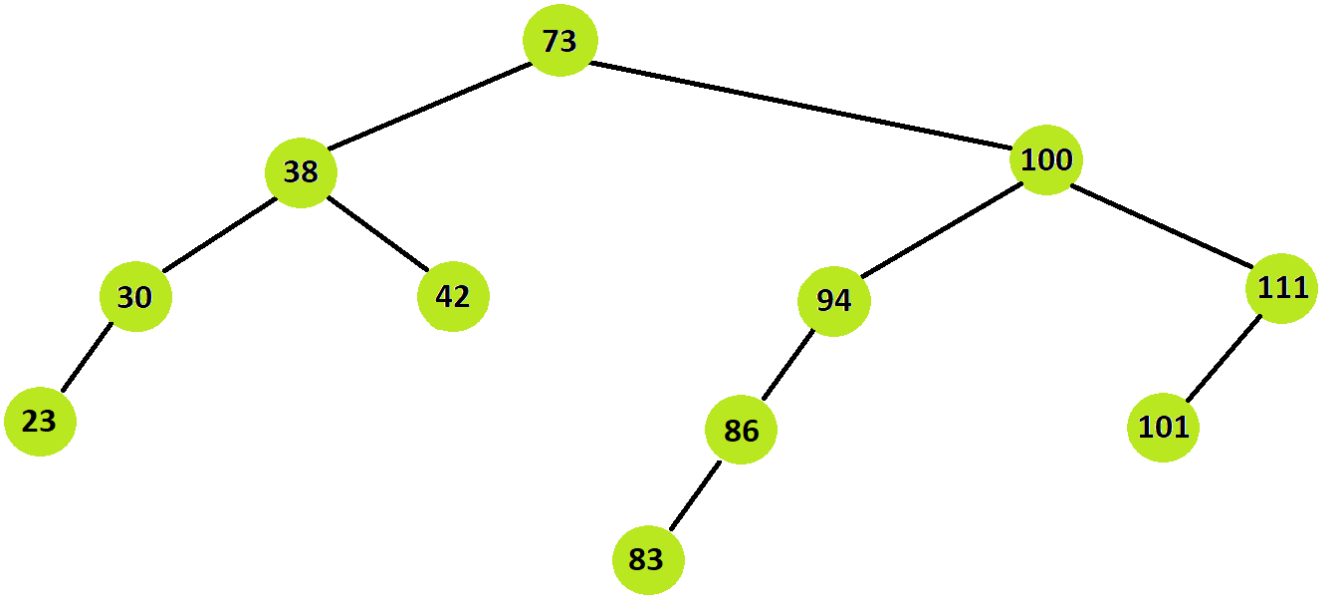
After deletion (63)



After deletion (33)



After deletion (53)



Question 4

addNgram

(n is count of the nodes in the tree)

(s is the size of the strings of the ngrams)

Worst case happens when the height of the tree is equal to the count of the nodes and the new node is not a duplicate. In other words, worst case happens when every node on the tree have one and only one child and the new node is not a duplicate. The new node should have an item that will increase the height of the tree for worst case to happen. It should not be a duplicate because the function must reach the leaf in its worst case. Size of the strings of the ngrams are also an important factor in the analysis of the addNgram since comparisons are made. The addNgram function actually first calls a function named toLower which converts the letters of the input string to lower case letters. toLower function runs in O(s) time complexity. Even though it has no effect on the overall complexity I showed it in red at the last step of the analysis.

I did not include the O(1)s in the analysis below since the answer will be given in Big-O notation (they do not change the answer).

$T(0) = O(s)$ (node = new TreeNode(ngram))

$T(n) = T(n-1) + O(s)$

$T(n) = T(n-2) + 2O(s)$

$T(n) = T(n-3) + 3O(s)$

.

.

.

$T(n) = T(n-k) + kO(s)$

when $k = n$

$T(n) = T(0) + nO(s)$

$T(n) = O(s) + nO(s)$

$T(n) = O(s) + O(sn)$ + O(s) (toLower)

Answer: O(sn)

operator<<

This function is not searching for a node. This function has to process every node in the tree once and only once. Hence, this function does not have different cases. It will always visit every single node only and only once. Size of the strings also have an impact when printing the ngrams. So, the complexity of this function is O(sn) where s is the size of the strings of the ngrams and n is the count of the nodes.

Answer: O(sn)

Sample Output

```
-bash-4.2$ g++ *.cpp -o hw2
-bash-4.2$ ./hw2 input.txt 4

Total 4-gram count: 6
"ampl" appears 1 time(s)
"hise" appears 1 time(s)
"mple" appears 1 time(s)
"samp" appears 1 time(s)
"text" appears 1 time(s)
"this" appears 2 time(s)

4-gram tree is complete: No
4-gram tree is full: No

Total 4-gram count: 8
"aatt" appears 1 time(s)
"ampl" appears 1 time(s)
"hise" appears 1 time(s)
"mple" appears 1 time(s)
"samp" appears 3 time(s)
"text" appears 1 time(s)
"this" appears 2 time(s)
"zinc" appears 1 time(s)

4-gram tree is complete: No
4-gram tree is full: No
-bash-4.2$
```