Trees

Maryam Shahid

21801344

CS202

Assignment 2

Question 1

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

Infix: A x B x C + D – E – F / G + H

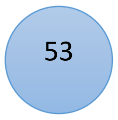
Postfix: A B x C D + E – x F G H + / -

Question 2

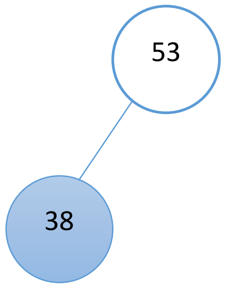
Insertion:

Initially: no node present; Tree is empty

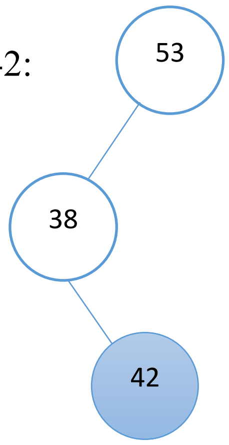
Insert 53:



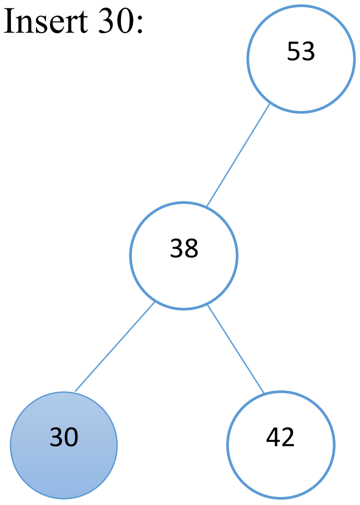
Insert 38:



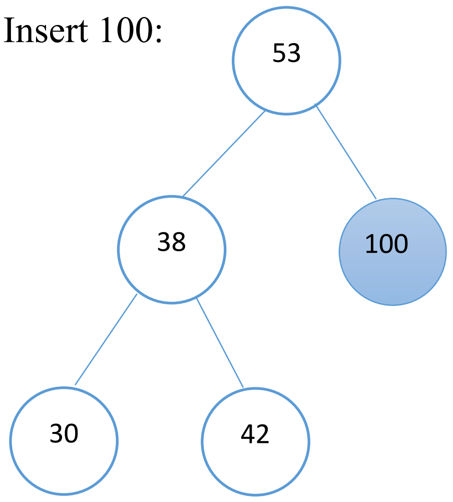
Insert 42:



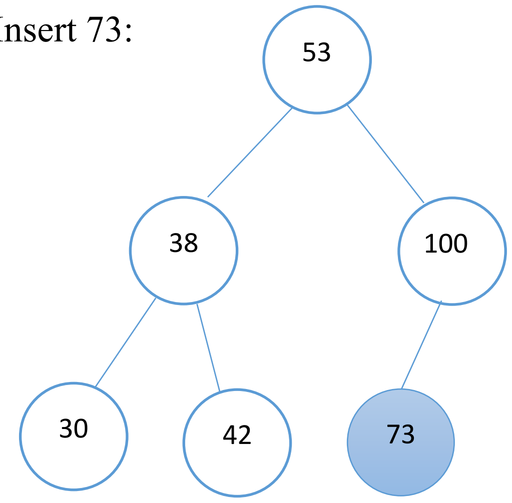
Insert 30:



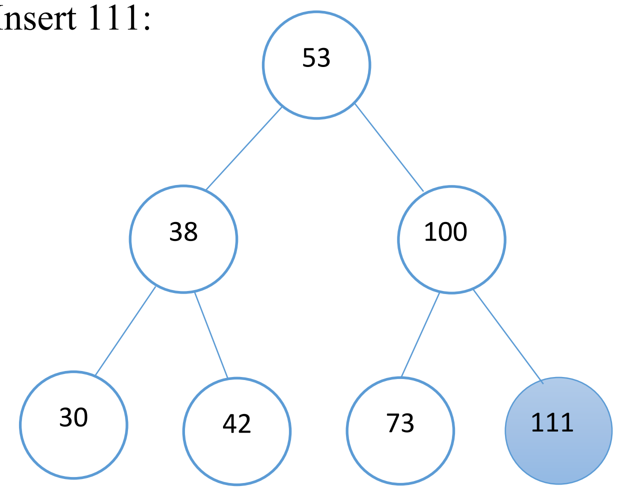
Insert 100:



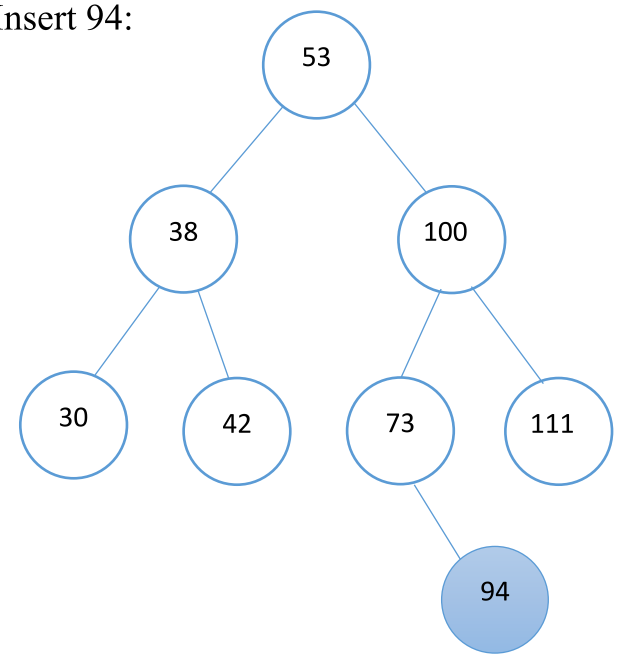
Insert 73:



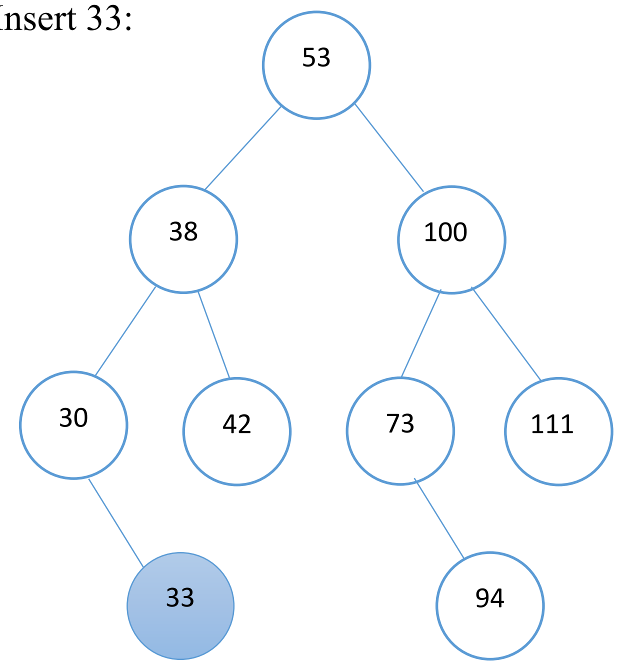
Insert 111:



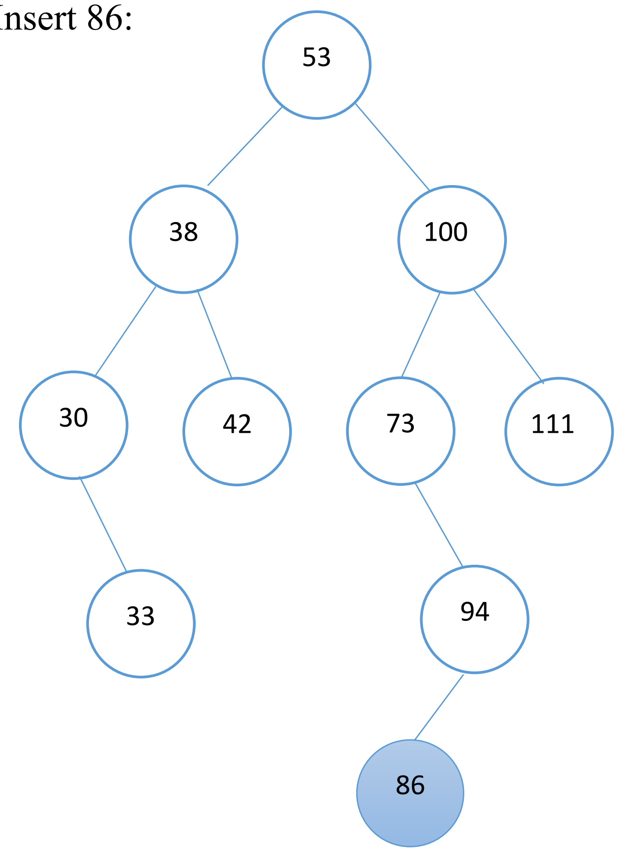
Insert 94:



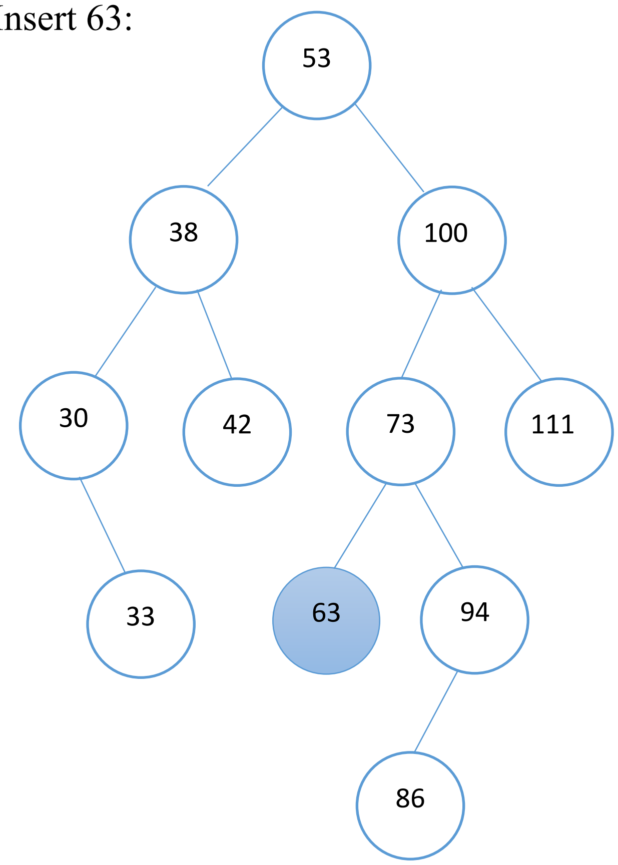
Insert 33:



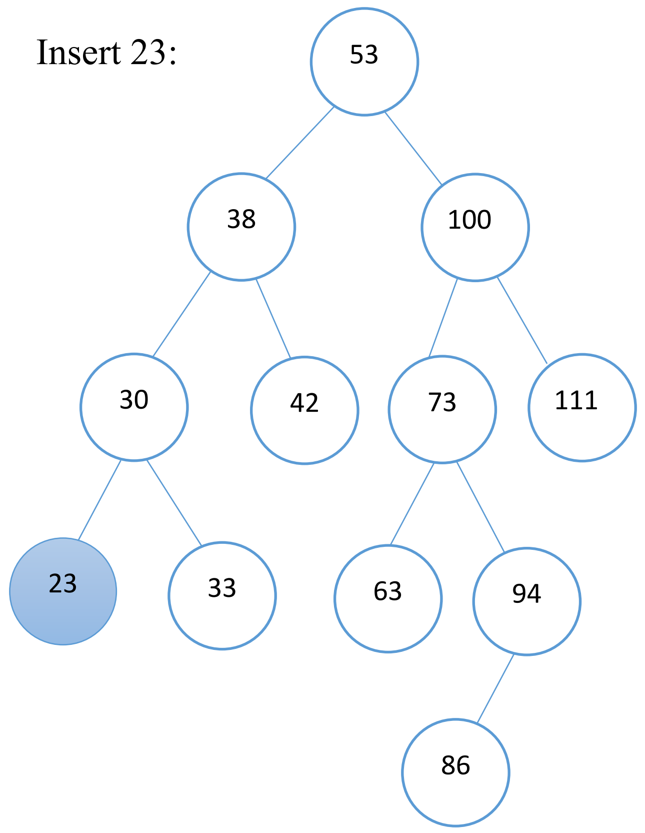
Insert 86:



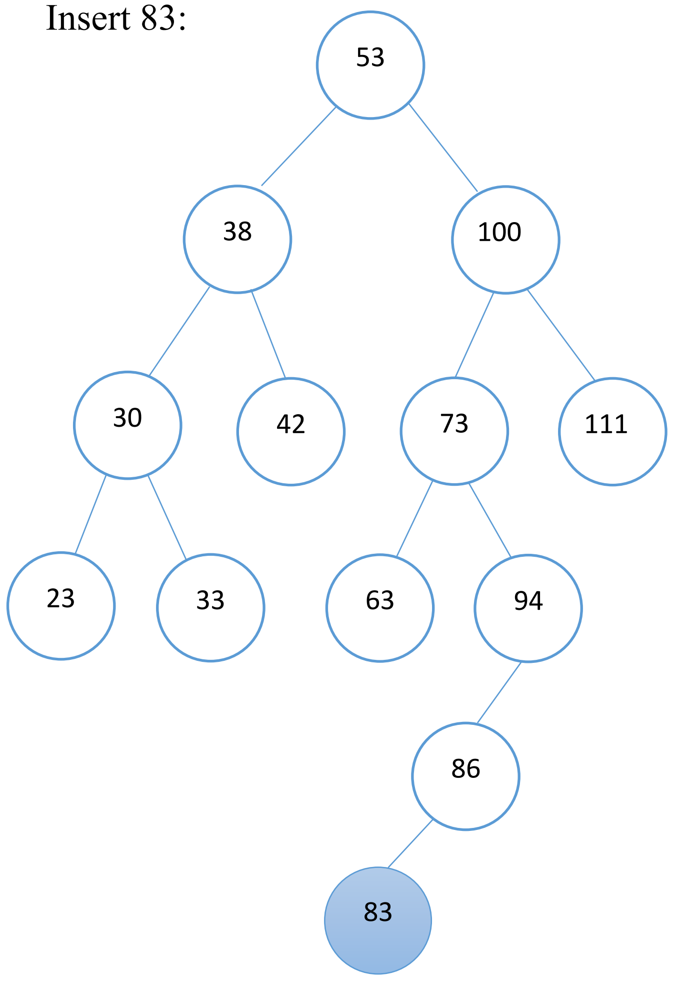
Insert 63:



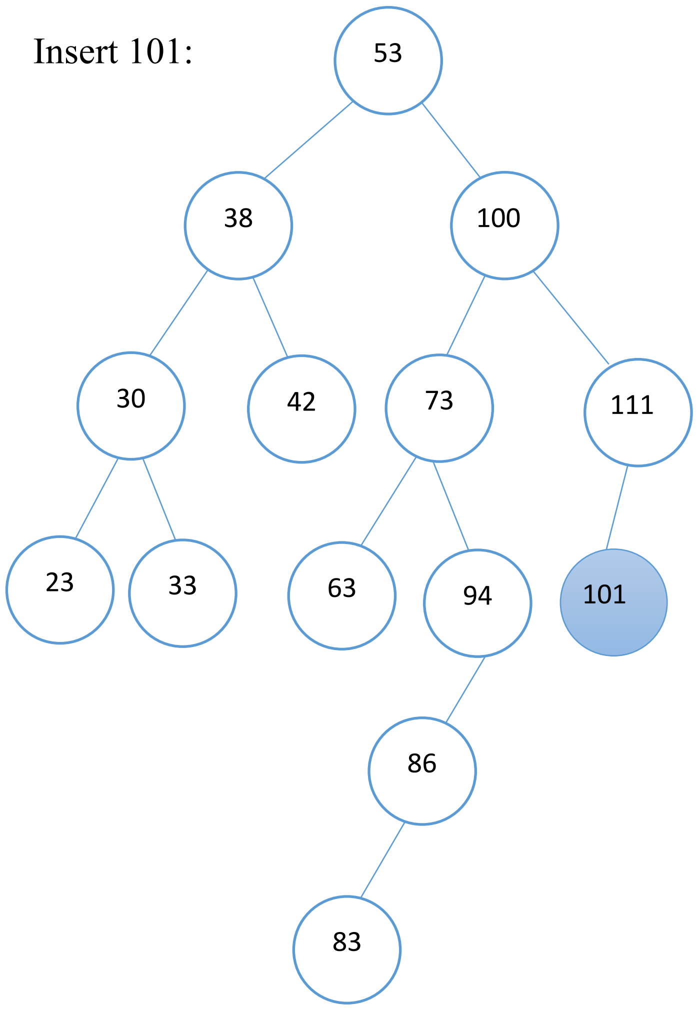
Insert 23:



Insert 83:

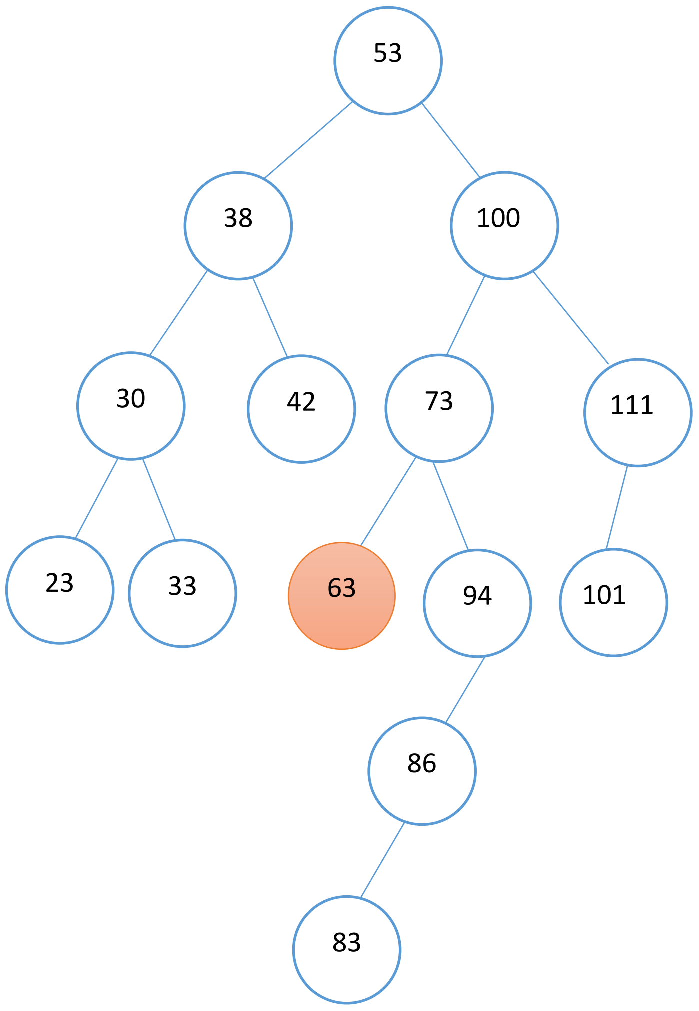


Insert 101:

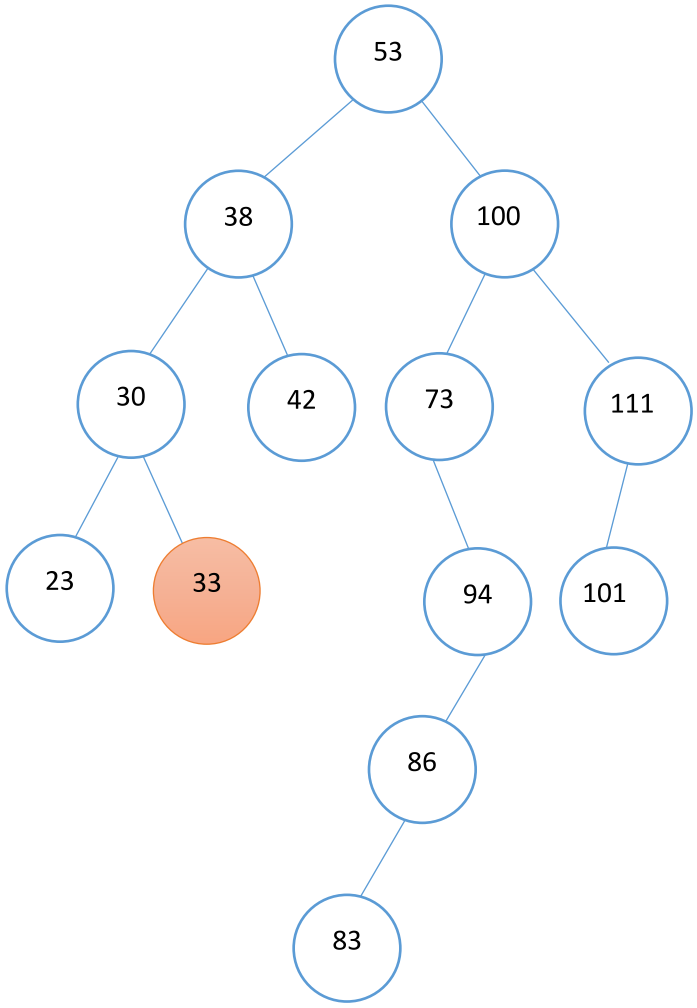


Deletion

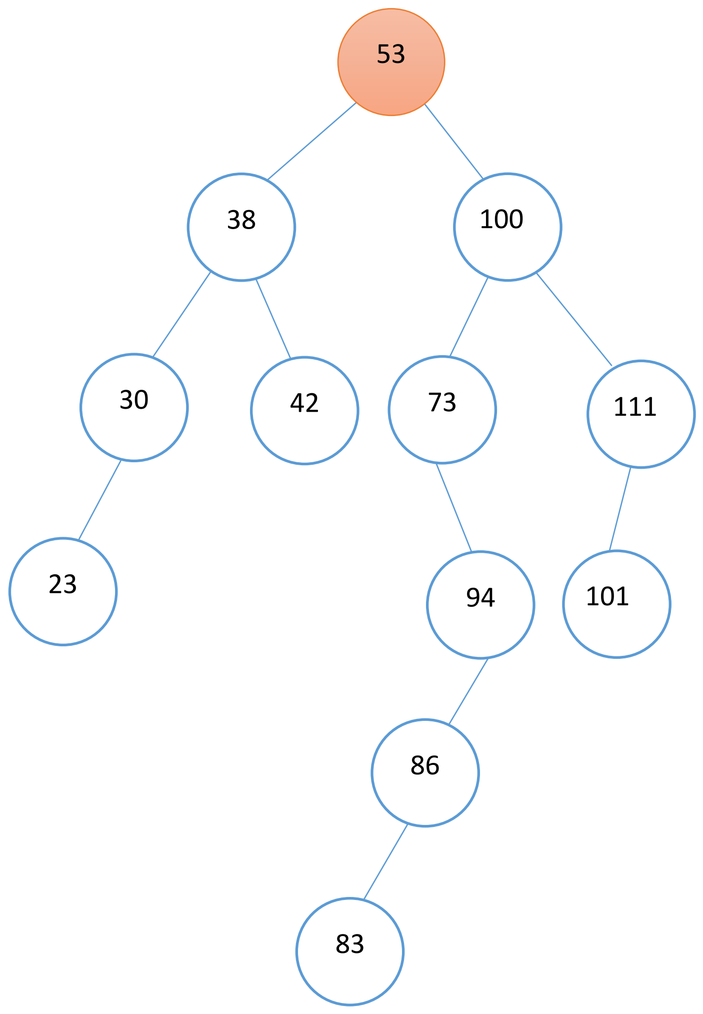
Delete 63:



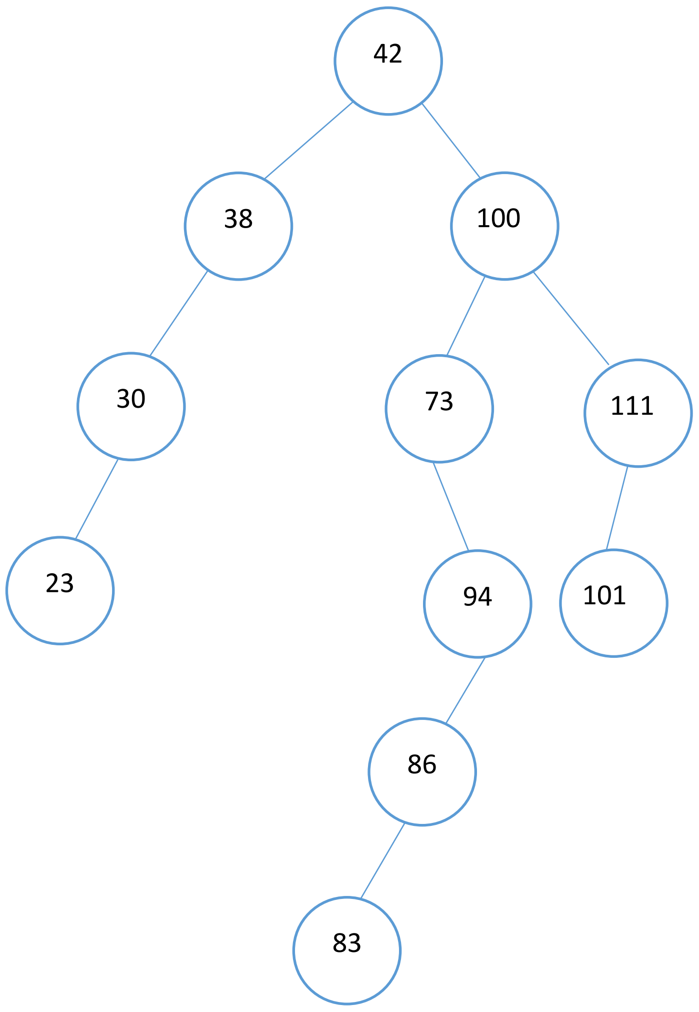
After deleting 63, delete 33:



After deleting 33, delete 53:



After deleting 53:



Question 4 - Worst case running time complexities of:

addNgram:

The pseudocode for the algorithm is as follows:

```
addNgram (string: ngram)
        bring all letters of ngram to lowercase
        if (tree is empty)
            create a new node and set ngram as the root pointer
        else if (the node is already present)
            check if ngram node is already present using searchNode
            increment ngram count
        else
            insert the new ngram node where searchNode ended

searchNode( string: ngram)
        // pre-order traversal of tree
        if  (rootPtr is not empty)
            if  (input ngram is smaller than ngram of rootPtr)
                recursive call to searchNode on left subtree of rootPtr
            else
            // input is larger than ngram of rootPtr
                recursive call to searchNode on right subtree of rootPtr
        return node if present
```

The addNgram function adds a node to the binary search tree. When the function is called, it checks whether the tree is empty, the time taken for this is constant. If the tree is not empty, a helper function, searchNode is called. As the name suggests, this function searches the tree for the node to be inserted. It persons an in-order traversal on the tree. In the worst case, the entire tree would have to be traversed to visit each node. The time complexity would depend on the height of the tree. However, in the worst case, the height is n-1 so this gives O(n). In case the node is present, no new node is inserted and only the ngramcount is incremented. In case the node is not already presented, the new node is inserted where the searchNode function ended which has a constant time complexity. Hence, in the worst case this function would result in the time complexity of O(n).

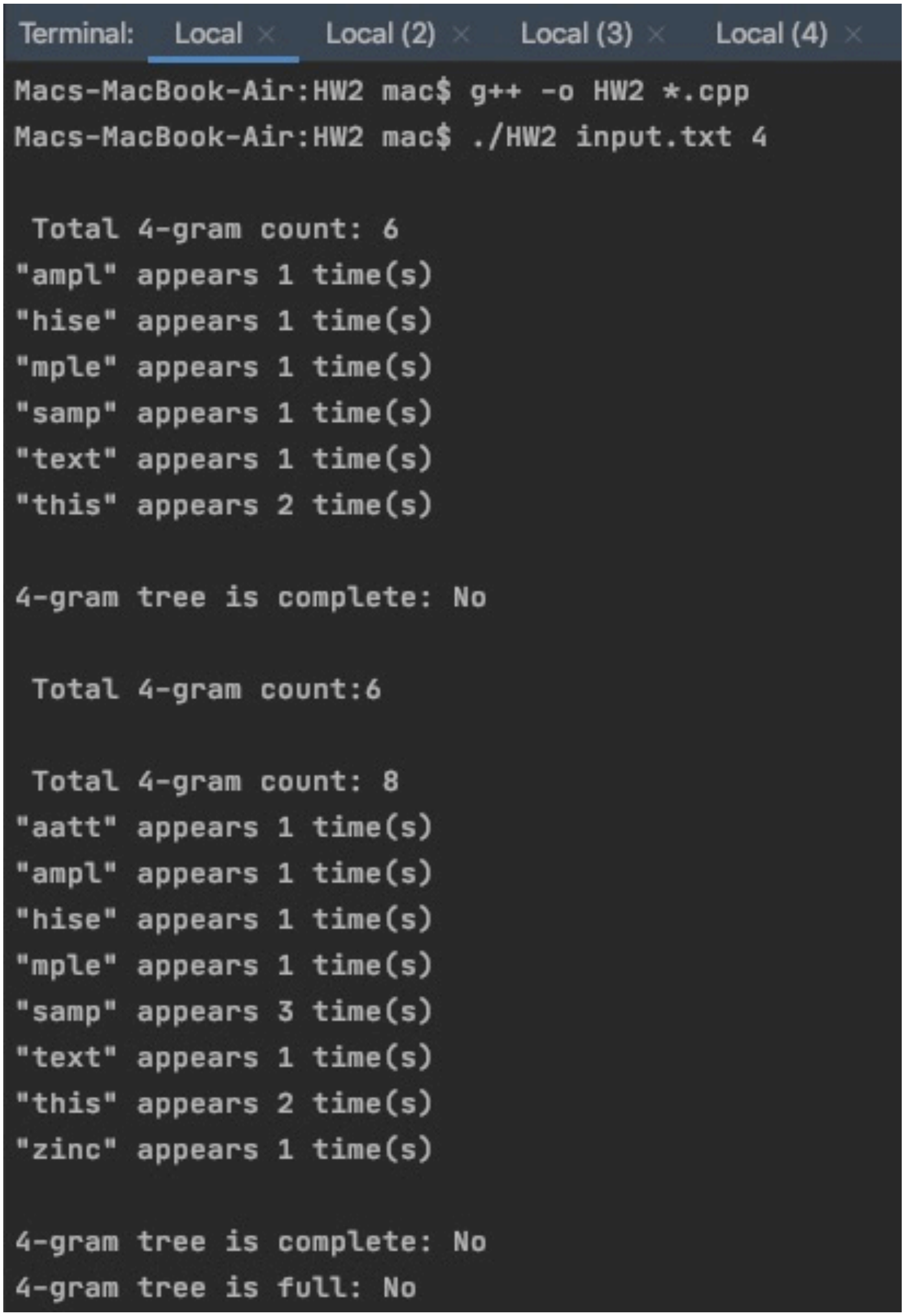
operator<<:

The pseudocode for the algorithm is as follows:

```
operator << (ostream: out, NgramTree: tree)
        call output helper on tree
        return output helper's output
output helper (<<(ostream: out, NgramNode*: rootPtr)
        // in-order traversal of the tree
        recursive call to output helper on rootPtr's left subtree
        visit rootPtr
        recursive call to output helper on rootPtr's right subtree
```

As noted from above, the main work is done by the output helper which performs an in-order traversal on the nodes present in the binary tree. If there are n nodes present in the tree then the worst case time complexity by in-order travel is O(n). Since the work done by the operator<< function itself is a call to output helper, its time complexity for the call is constant. Therefore the worst case time complexity of this function is O(n).

Example Screenshot:



```
Macs-MacBook-Air:HW2 mac$ g++ -o HW2 *.cpp
Macs-MacBook-Air:HW2 mac$ ./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

 Total 4-gram count:6

 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
```