

HOMEWORK 02

Solution for Question 01:

prefix expression:

without parantheses: $- x \times A B - + C D E / F + G H$

with parantheses: $-(x(xAB)(-(+CDE))(/F)(+GH))$

infix expression:

without parantheses: $A \times B \times C + D - E - F / G + H$

with parantheses: $((A \times B) \times ((C + D) - (E))) - (F / (G + H))$

postfix expression:

without parantheses: $AB \times CD + E - x FGH + / -$

with parantheses: $((AB \times) ((CD +) E -) x) (F (GH +) /) -$

Solution for Question 02:

We are asked to draw the initially empty Binary Search Tree after operations as follows :

insert 53, 38, 42, 30, 100, 73, 111, 94, 33, 86, 63, 23, 83, 101; then delete 63, 33, 53.

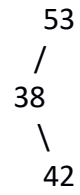
insert(53):

53

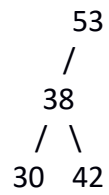
insert(38):

53
/
38

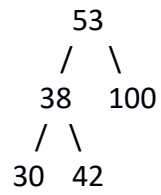
insert(42):



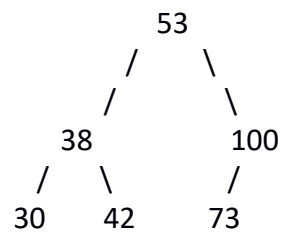
insert(30):



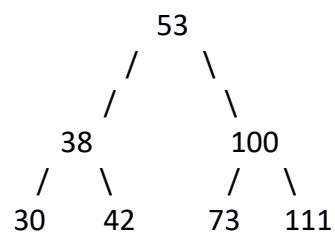
insert(100):



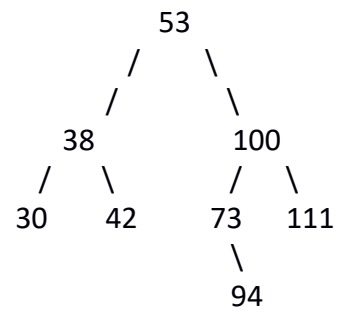
insert(73):



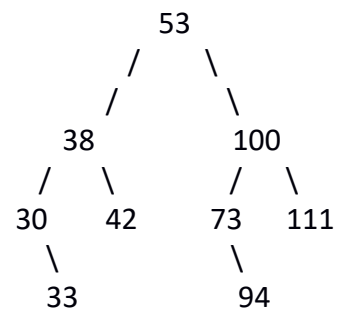
insert(111):



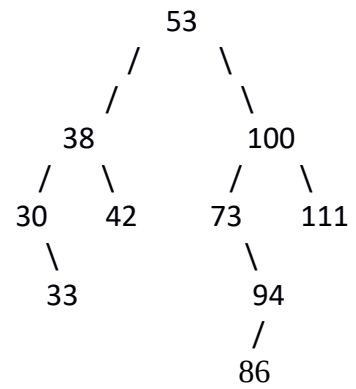
insert(94):



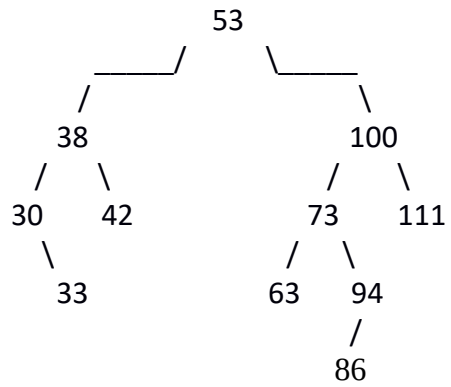
insert(33):



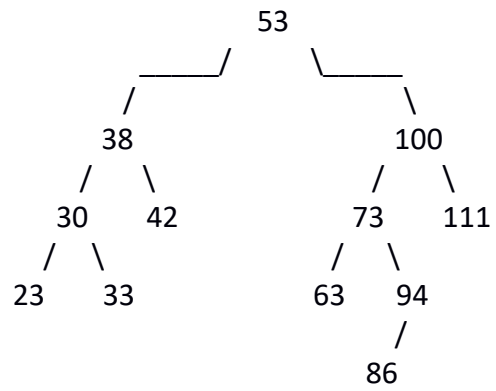
insert(86)



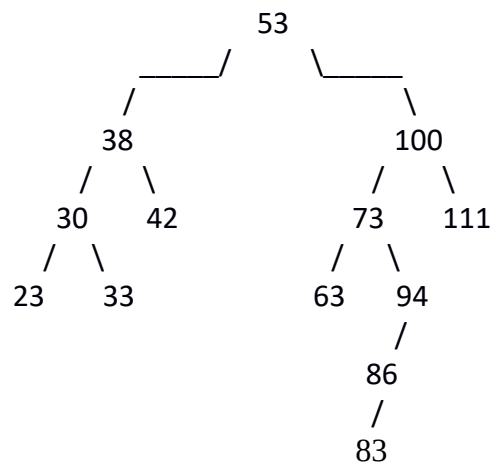
insert(63)



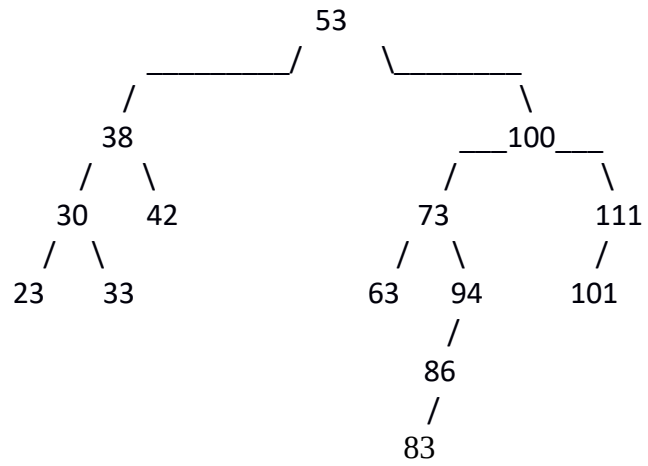
insert(23)



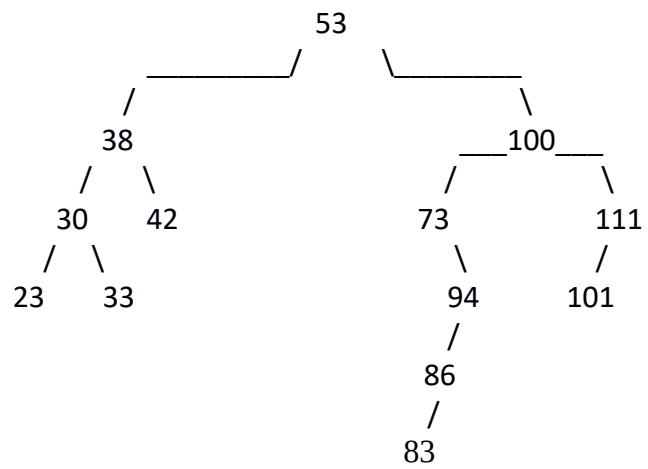
insert(83)



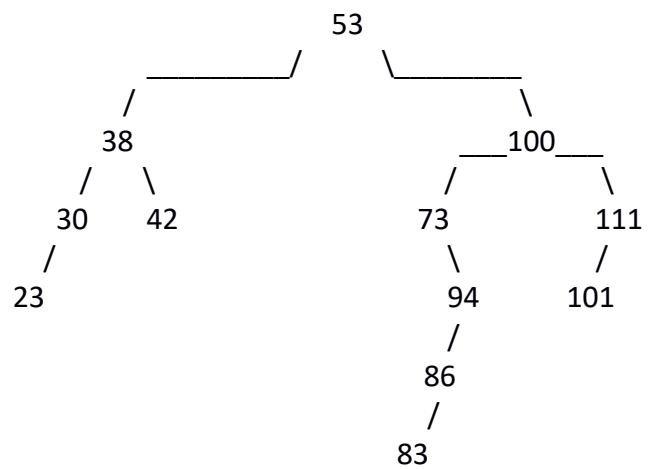
insert(101)



delete(63)

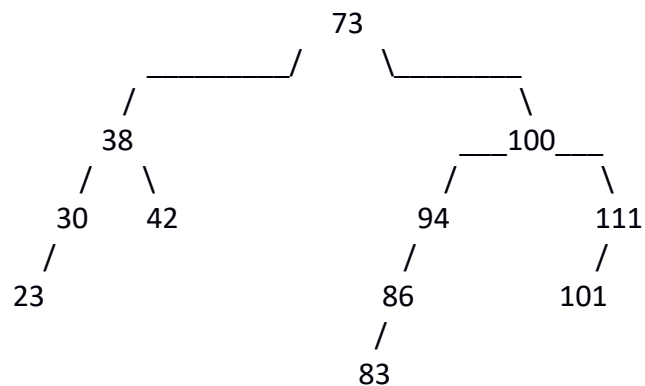


delete(33)



delete(53)

→ take the leftmost node of the right child of 53 and then put it in 53's place after deleting 53. ←



Sample output of the Program for Question 03:

```
g++ -o deneme *.cpp; ./deneme input.txt 4
```

Total 4-gram count: 6

"ampl" appears 1 times.

"hise" appears 1 times.

"mple" appears 1 times.

"samp" appears 1 times.

"text" appears 1 times.

"this" appears 2 times.

4-gram tree is complete: No

4-gram tree is full: No

Total 4-gram count: 8

"aatt" appears 1 times.

"ampl" appears 1 times.

"hise" appears 1 times.

"mple" appears 1 times.

"samp" appears 3 times.

"text" appears 1 times.

"this" appears 2 times.

"zinc" appears 1 times.

4-gram tree is complete: No

4-gram tree is full: No

Solution for Question 4:

We are asked to analyze the worst-case running time complexities of the **addNgram** and **operator<<** functions in the question03 using the big-O notation.

Analysis of addNgram:

Worst case for adding an n-gram into the list occur when the sequence of already sorted elements (which all are either greater or lesser than the n-gram that we want to insert) are added to the tree repeatedly. Without loss of generality, we can say that if we add 1,2,3...,n into a tree, we will get a tree with height of n-1. Without loss of generality, let's assume that ngram that we want to insert is greater than all of the elements in the tree.

Then:

$T(n) = 1$ when $n \leq 1$
and otherwise we have $T(n) = T(n-1) + O(1)$

$$\begin{aligned} T(n) &= T(n-1) + O(1) \\ &= T(n-2) + O(1) + O(1) \\ &= T(1) + O(1) + \dots + O(1) \\ &= nO(1) = O(n) \end{aligned}$$

So, worst-case complexity of **addNgram** is $O(n)$.

Analysis of operator<<:

In the **operator<<** method, I directly make a call to **inOrderTraversal** so the worst-case complexity of **operator<<** will be equal to the worst-case complexity of **inOrderTraversal** method. Regardless of the cases, the complexity of **inOrderTraversal** is **$O(n)$** since we iterate through each existing node only once.

So, worst-case complexity of **operator<<** is $O(n)$.